以数据分析和挖掘原理为基础，结合作者多年的互联网应用实践经验，将R语言和网站分析有机结合，包含大量R语言实施案例，力求做到“授之以渔”

数据分析
技术丛书

华章科技

Web Analysis and Mining With R

R语言与网站分析

李明/著



机械工业出版社
China Machine Press

R语言与网站分析

Web Analysis and Mining With R



在当下大数据时代，越来越多的网站开始重视数据，期望从数据中寻找有价值的结论，指导公司管理层的决策，最终创造更大的网站价值。而在各种网站分析技术中，R语言作为一个可进行交互式数据分析和探索的强大平台，起着举足轻重的作用，R语言的免费开源使得很多公司用它来处理数据、分析数据、完成模型。使用R语言可以进行网站数据系统的搭建，可以对累积的众多数据进行挖掘，找出其中的特征和规律。同时R语言是目前包含数据算法最全的工具，其灵活强大的绘图功能可以更好地表现数据处理结果。对于有志成为互联网数据挖掘/分析师的读者来说，掌握R语言将成为他们未来必备的技能之一。本书将R语言和网站分析/挖掘有机地结合起来，通过对大量案例进行详细阐述和深入分析，进而指导读者在实际工作中的应用。

本书的主要内容和特色：

- 本书包括大量的实际应用案例以及全部的R语言实现代码，通过对应用场景以及数学原理的分析，力图使读者在阅读过本书之后达到举一反三的效果。
- 本书在阐述实际案例的同时给出了其原始数据集的结构、应用场景以及结论分析，做到从“目标变量的抽象”、“数据预处理”直至最后“模型性能评估”的全流程讲解。
- 在讲解R语言函数时书中会给出详细的参数说明表以及数据挖掘原理，使读者不仅知其然，更能知其所以然。书中对数学原理的阐述不仅仅对互联网从业者有用，对于非互联网行业的数据从业人员也是有益处的。
- 本书的灵魂是互联网数据应用，所以在案例讲解过程中均会加入结论性分析，进而帮助数据分析师提高“如何利用结果数据指导实际商务决策”的能力。
- 全书对于具有互联网行业特点的“关系网络分析”和“文本分析”也做了较为详细的讲解。

投稿热线：(010) 88379604
客服热线：(010) 88378991 88361066
购书热线：(010) 68326294 88379649 68995259

华章网站：www.hzbook.com
网上购书：www.china-pub.com
数字阅读：www.hzmedia.com.cn

上架指导：计算机/数据分析

ISBN 978-7-111-45971-2

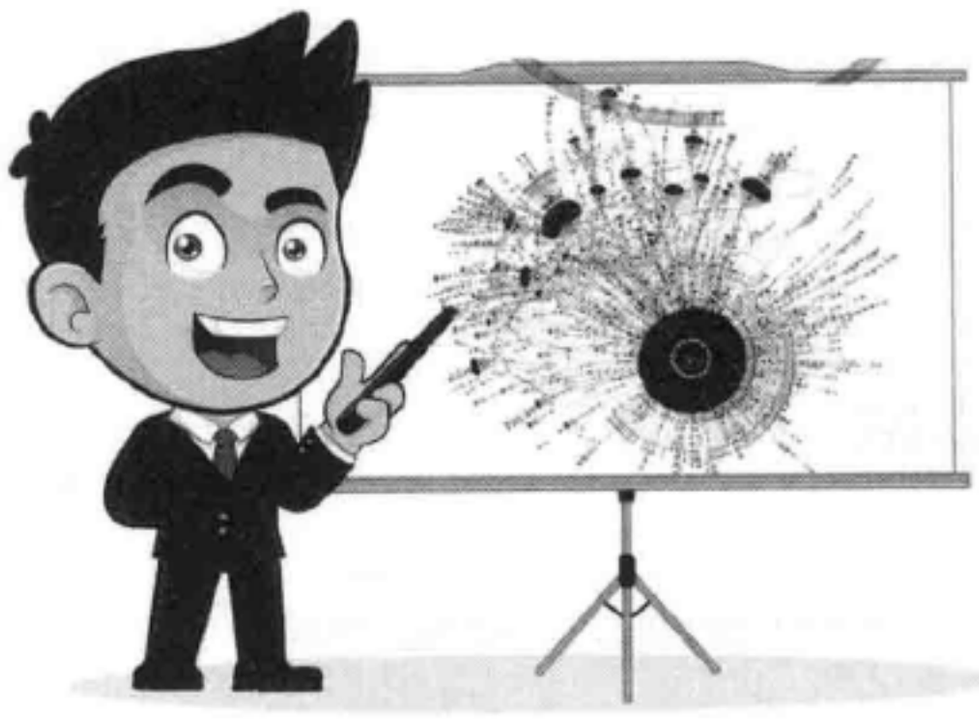
9 787111 459712 >

定价：79.00元

Web Analysis and Mining With R

R语言与网站分析

李明/著



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

R 语言与网站分析 / 李明著 . —北京：机械工业出版社，2014.4
（数据分析技术丛书）

ISBN 978-7-111-45971-2

I. R… II. 李… III. 程序语言 - 程序设计 IV. TP312

中国版本图书馆 CIP 数据核字（2014）第 033985 号

本书从互联网应用角度对 R 语言如何实现数据挖掘和指标分析等问题做了阐述。通过诸多真实应用案例的分析，作者试图为读者建立起一座沟通数学原理和互联网实际业务应用的桥梁。同时本书给出了案例中的完整代码以及分析过程，力图帮助读者充分理解 R 语言是如何实现算法的。

本书首先讲解 R 语言的基本语法和绘图方法；然后阐述 R 语言如何监控指标以及常用的网站分析方法；之后讲解互联网常用的数据挖掘方法：回归分析、分类分析、聚类分析、数据降维、关联分析、关系网络分析、文本分析等；最后讲解如何基于 R 语言建立一套 GUI 界面系统。

相信本书的内容对各位数据挖掘和互联网从业者都有较高的参考价值。

R 语言与网站分析

李 明 著

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：朱秀英
印 刷：北京市荣盛彩色印刷有限公司
开 本：186mm × 240mm 1/16
书 号：ISBN 978-7-111-45971-2
版 次：2014 年 4 月第 1 版第 1 次印刷
印 张：29.5（含 1 印张彩插）
定 价：79.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991 88361066
购书热线：（010）68326294 88379649 68995259
投稿热线：（010）88379604
读者信箱：hzjsj@hzbook.com

版权所有 · 侵权必究
封底无防伪标均为盗版
本书法律顾问：北京大成律师事务所 韩光 / 邹晓东

华章计算机
HZBOOKS | Computer Science and Technology

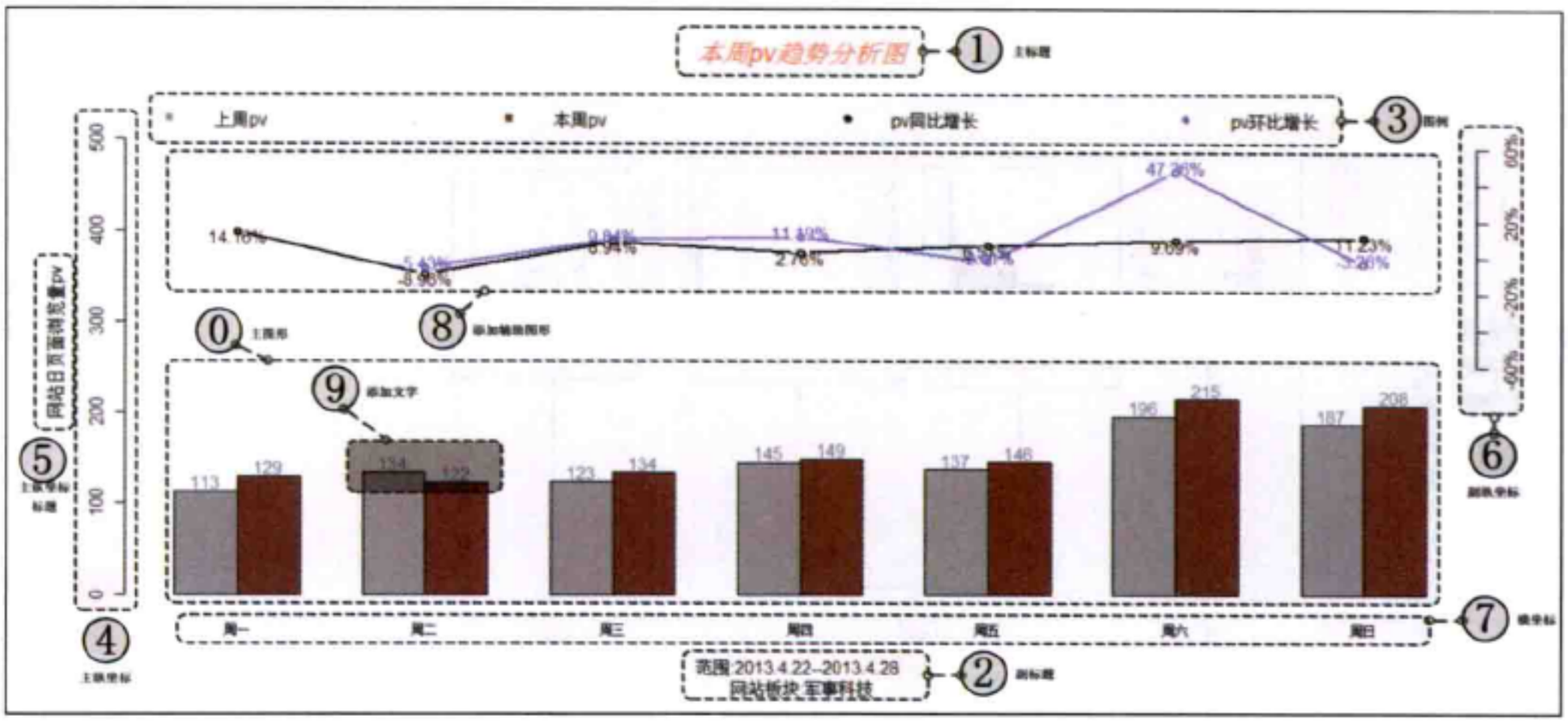


图 3-1　网站 PV 趋势图

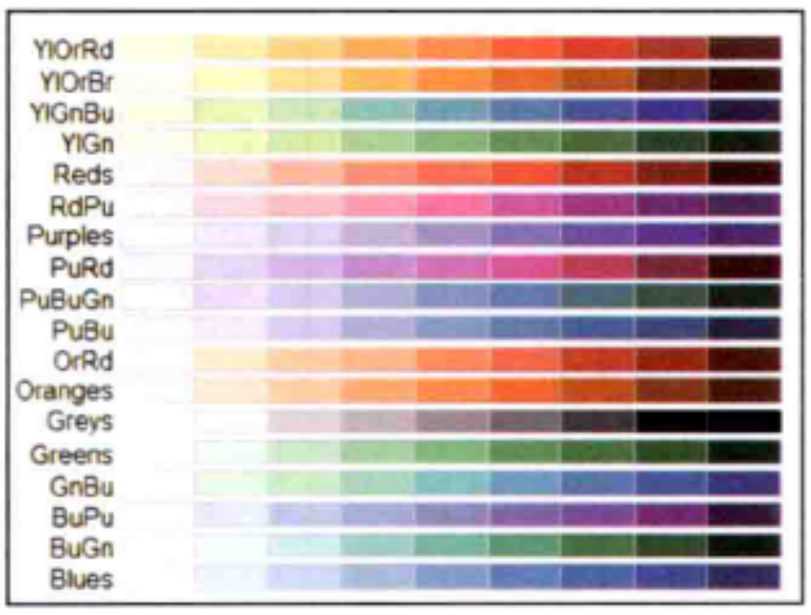


图 3-4　seq 连续型配色方案中各组颜色的名称和样式

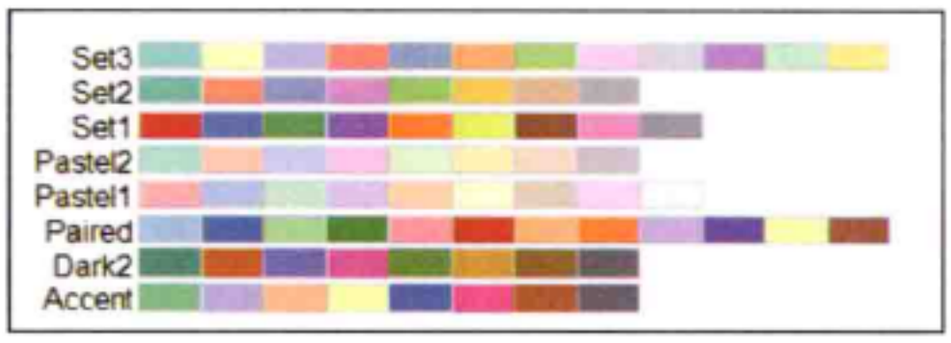


图 3-8　qual 离散型配色方案中各组的颜色

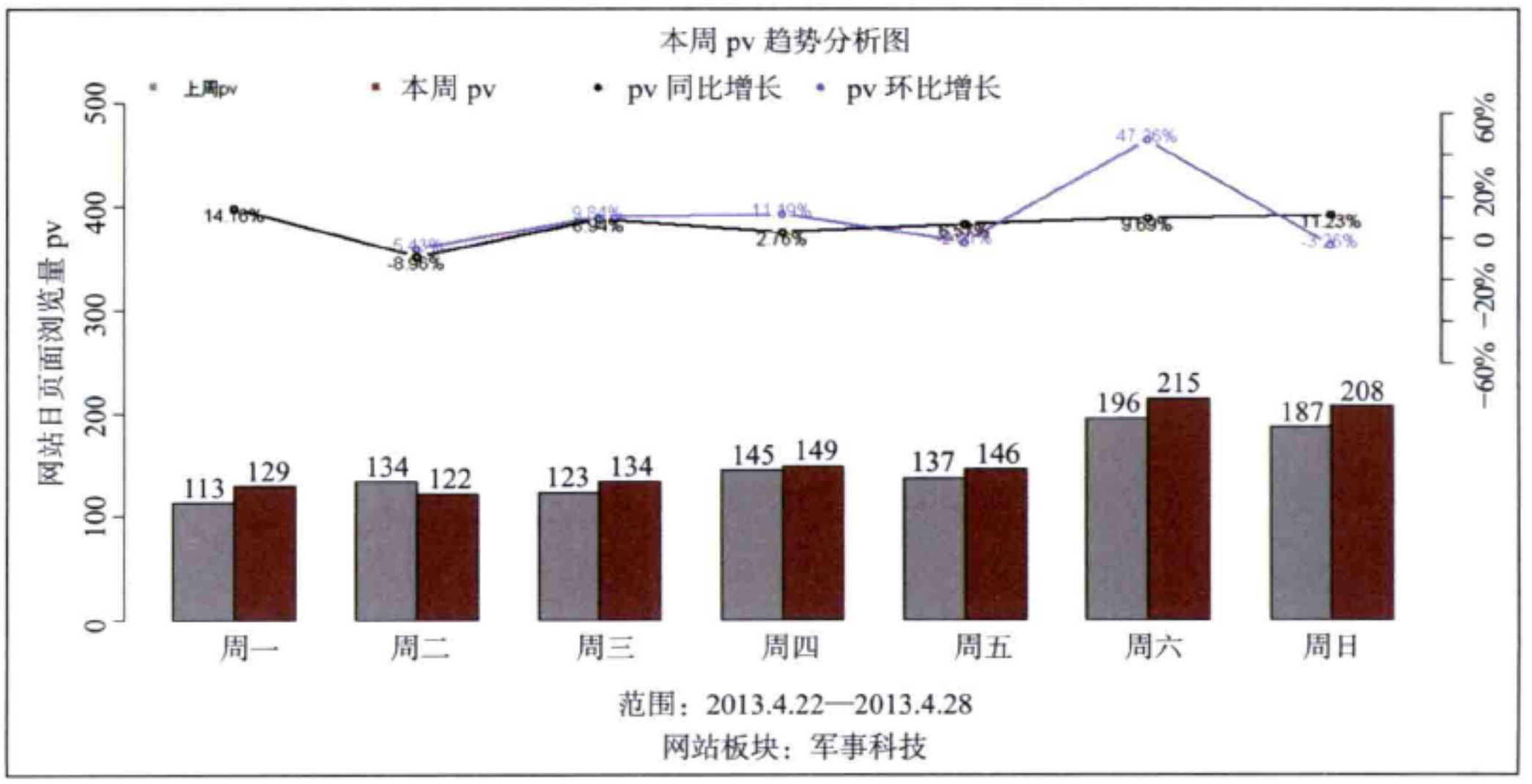


图 3-30　网站 PV 引例图 6

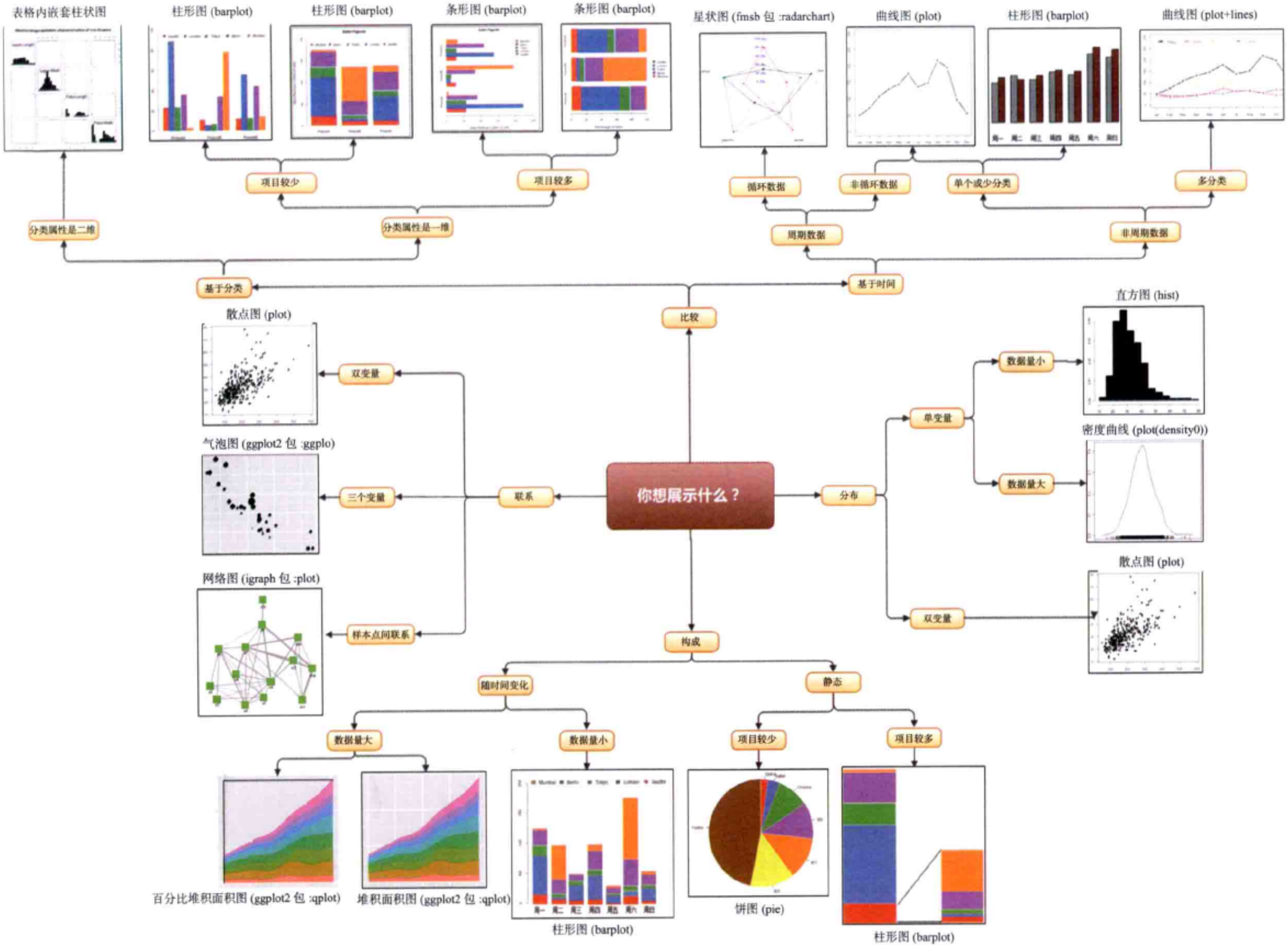

你想展示什么？
比较
联系
分布
构成
基于分类
基于时间
分类属性是二维
分类属性是一维
项目较少
项目较多
表格内嵌套柱状图
柱形图 (barplot)
柱形图 (barplot)
条形图 (barplot)
条形图 (barplot)
星状图 (fmsb 包 :radarchart)
曲线图 (plot)
柱形图 (barplot)
曲线图 (plot+lines)
循环数据
非循环数据
单个或少分类
多分类
周期数据
非周期数据
散点图 (plot)
双变量
气泡图 (ggplot2 包 :ggplo)
三个变量
网络图 (igraph 包 :plot)
样本点间联系
单变量
数据量小
数据量大
直方图 (hist)
密度曲线 (plot(density0))
双变量
散点图 (plot)
随时间变化
静态
数据量大
数据量小
项目较少
项目较多
百分比堆积面积图 (ggplot2 包 :qplot)
堆积面积图 (ggplot2 包 :qplot)
柱形图 (barplot)
饼图 (pie)
柱形图 (barplot)

图 3-32　图形选择思路

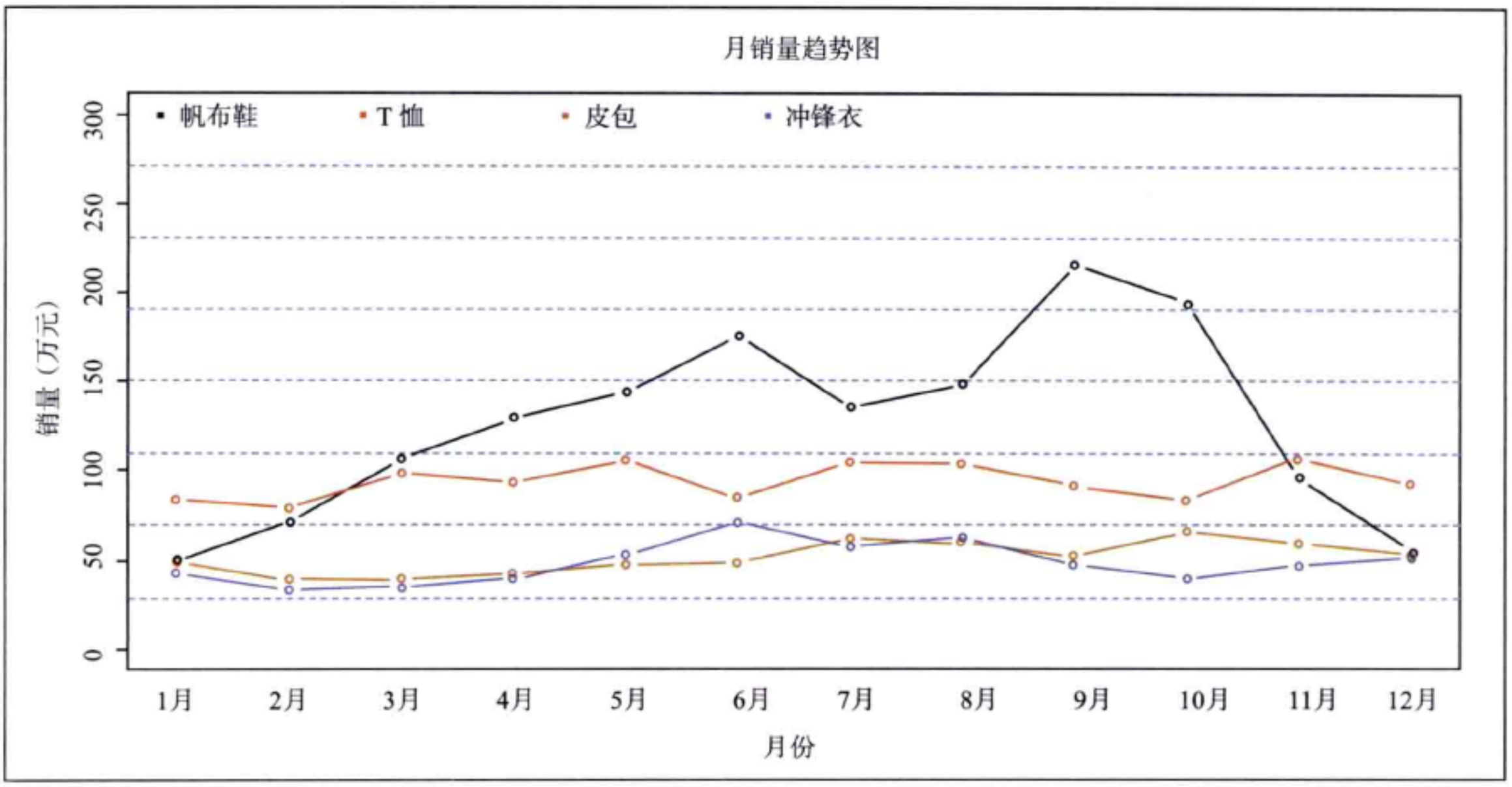


图 3-36　月销量趋势图

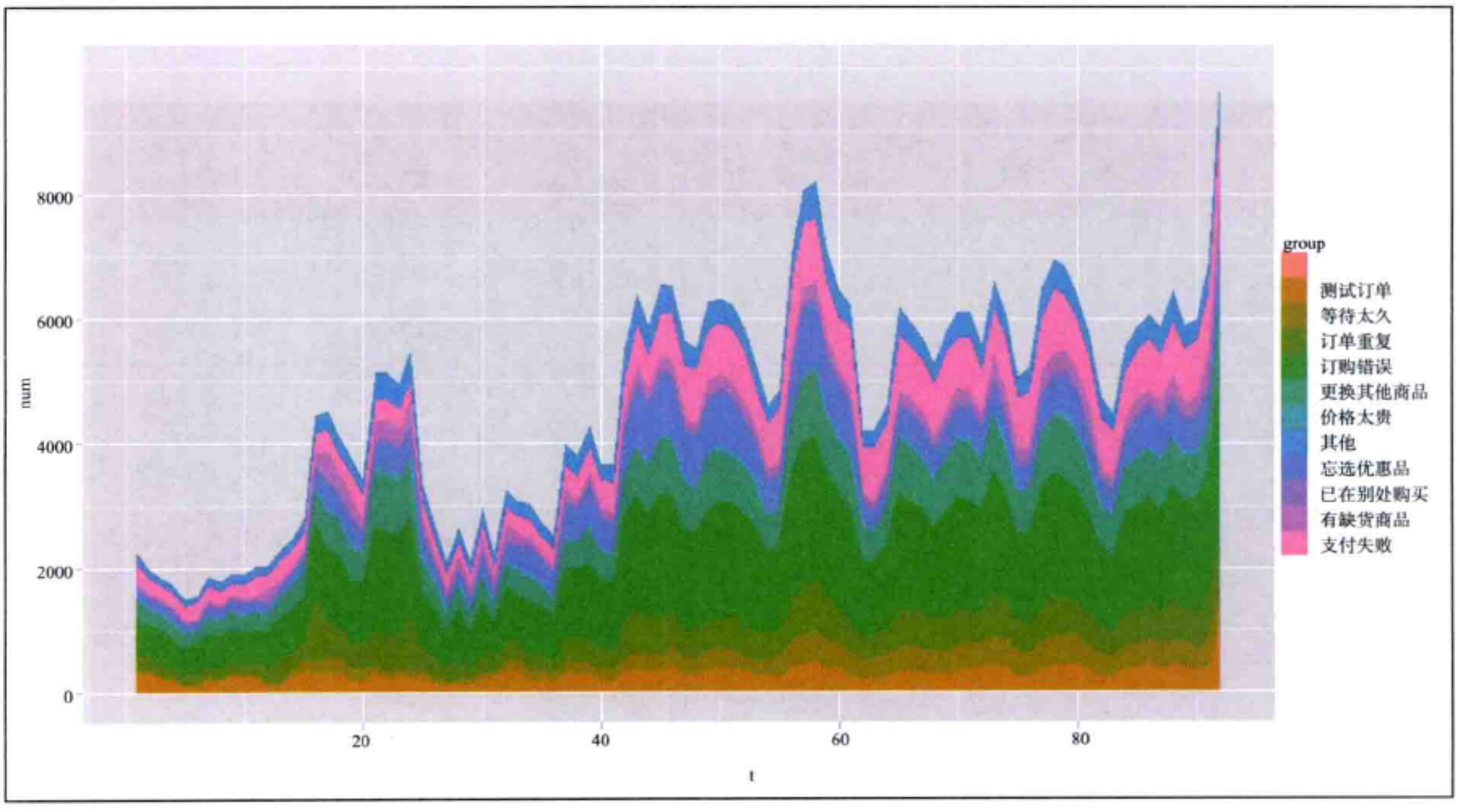


图 3-47　取消订单原因的面积堆积图

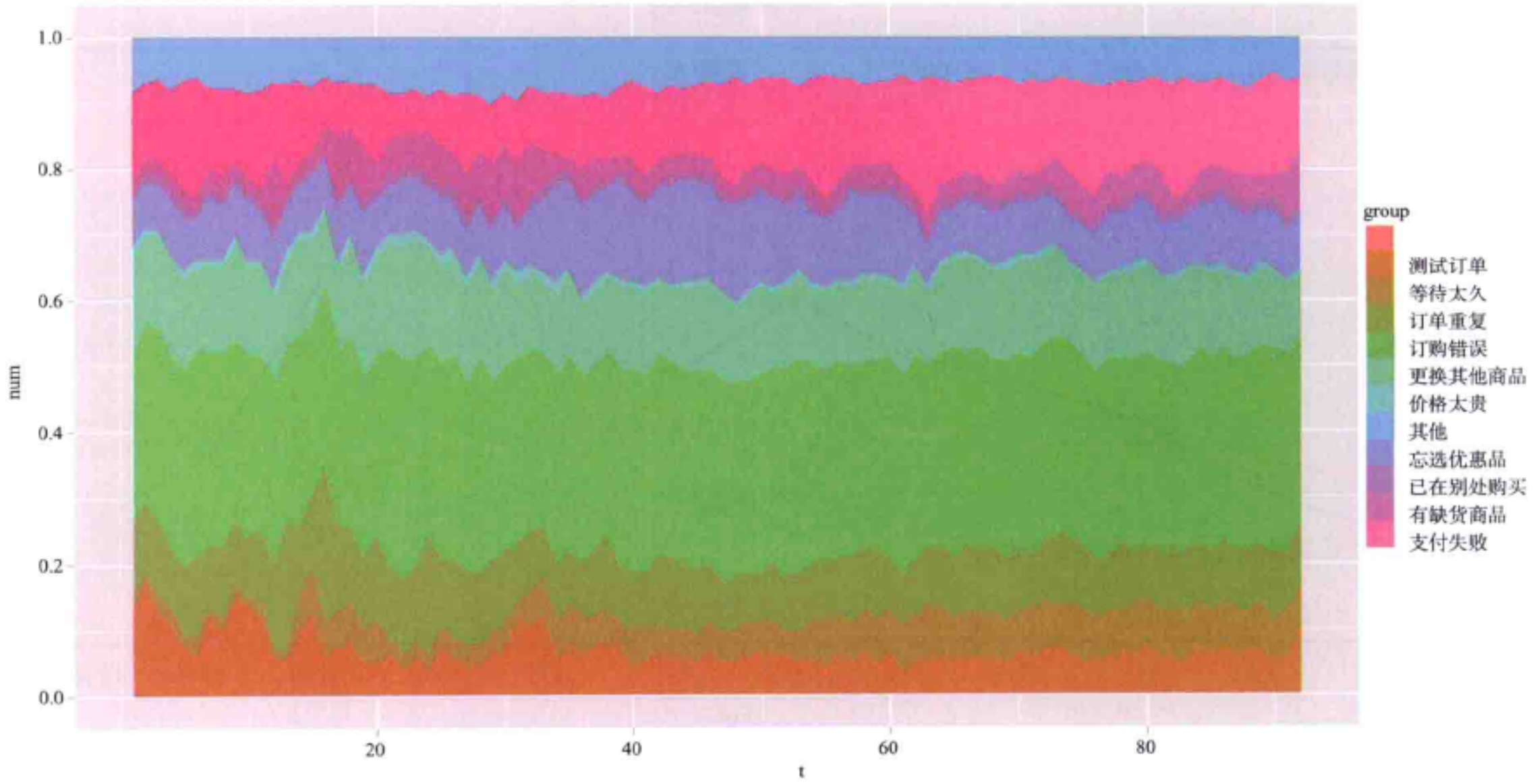


图 3-49　取消订单原因的面积堆积图

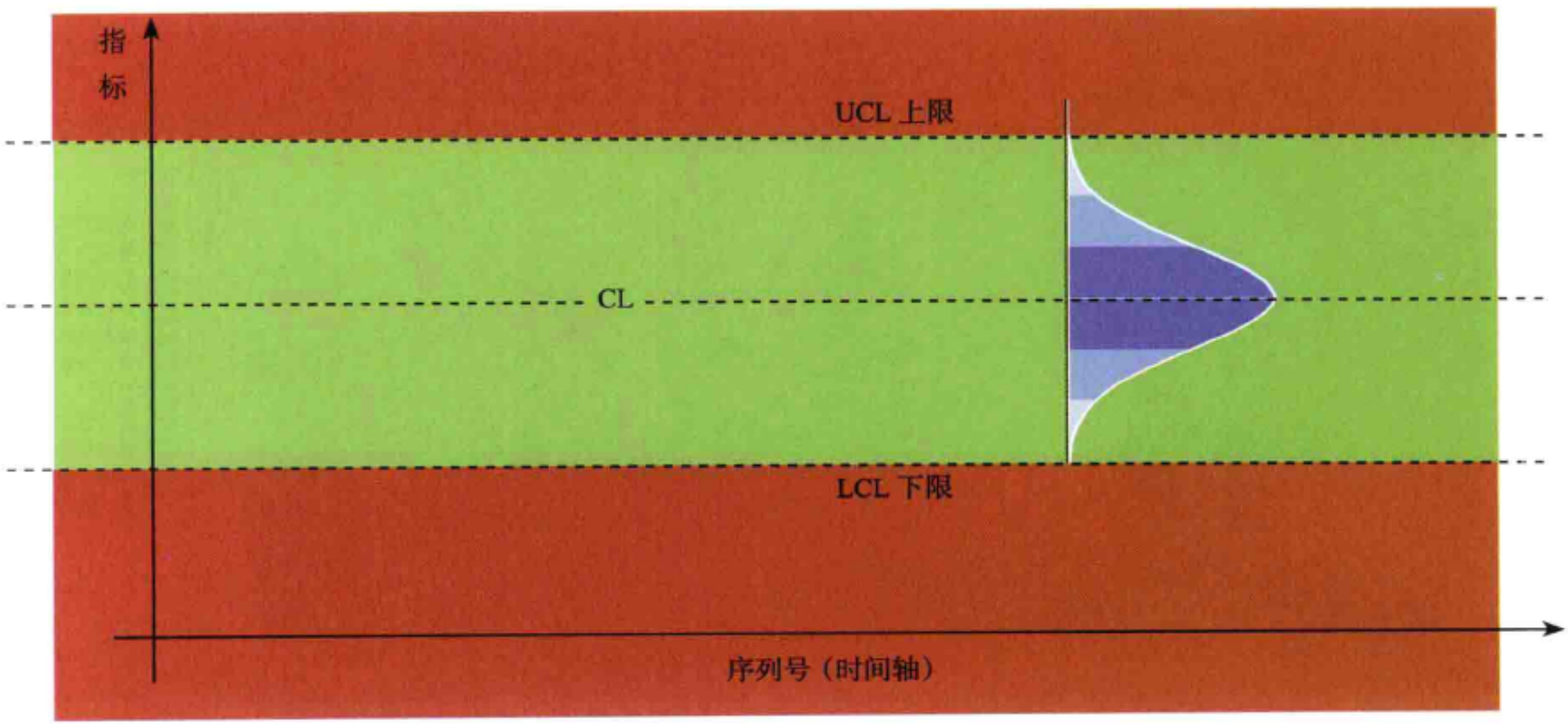


图 4-15　示例图

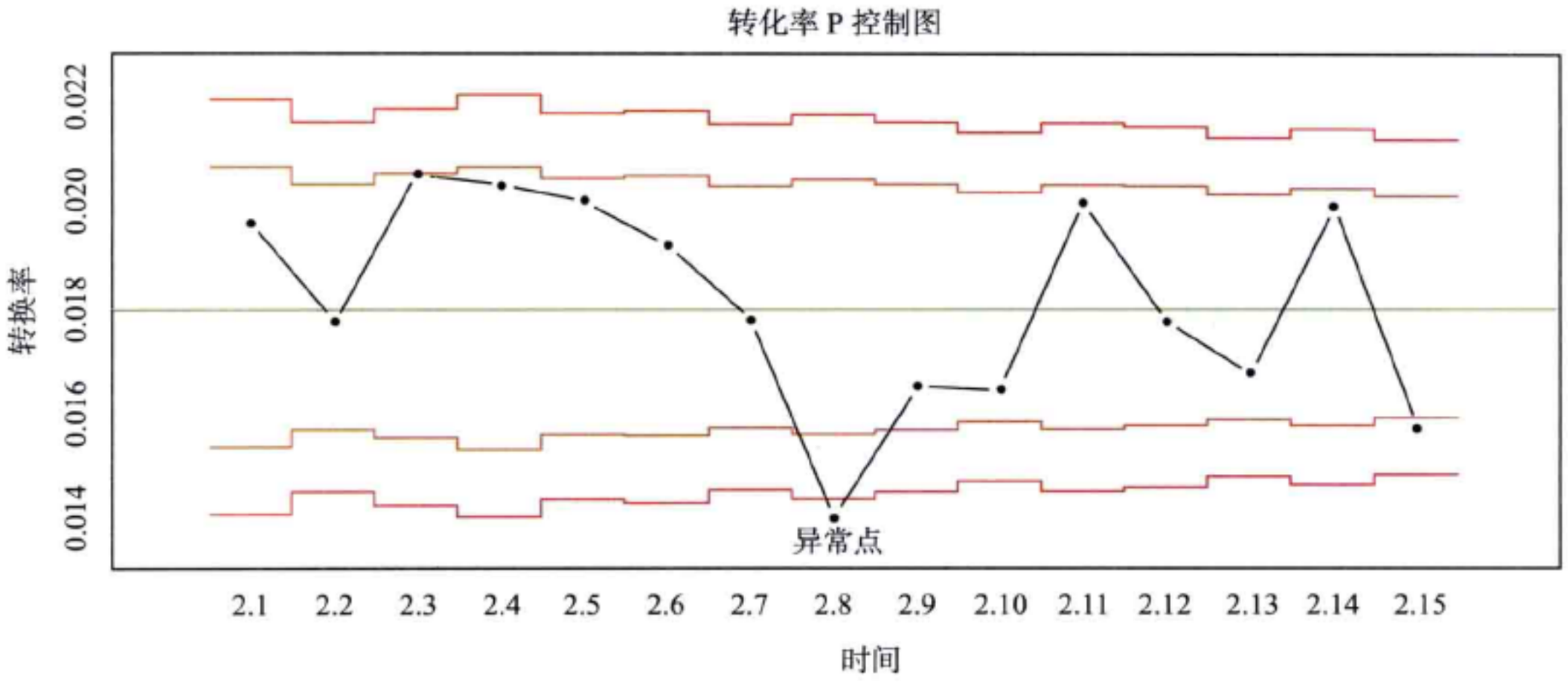


图 4-17　转化率指标的 P 控制图

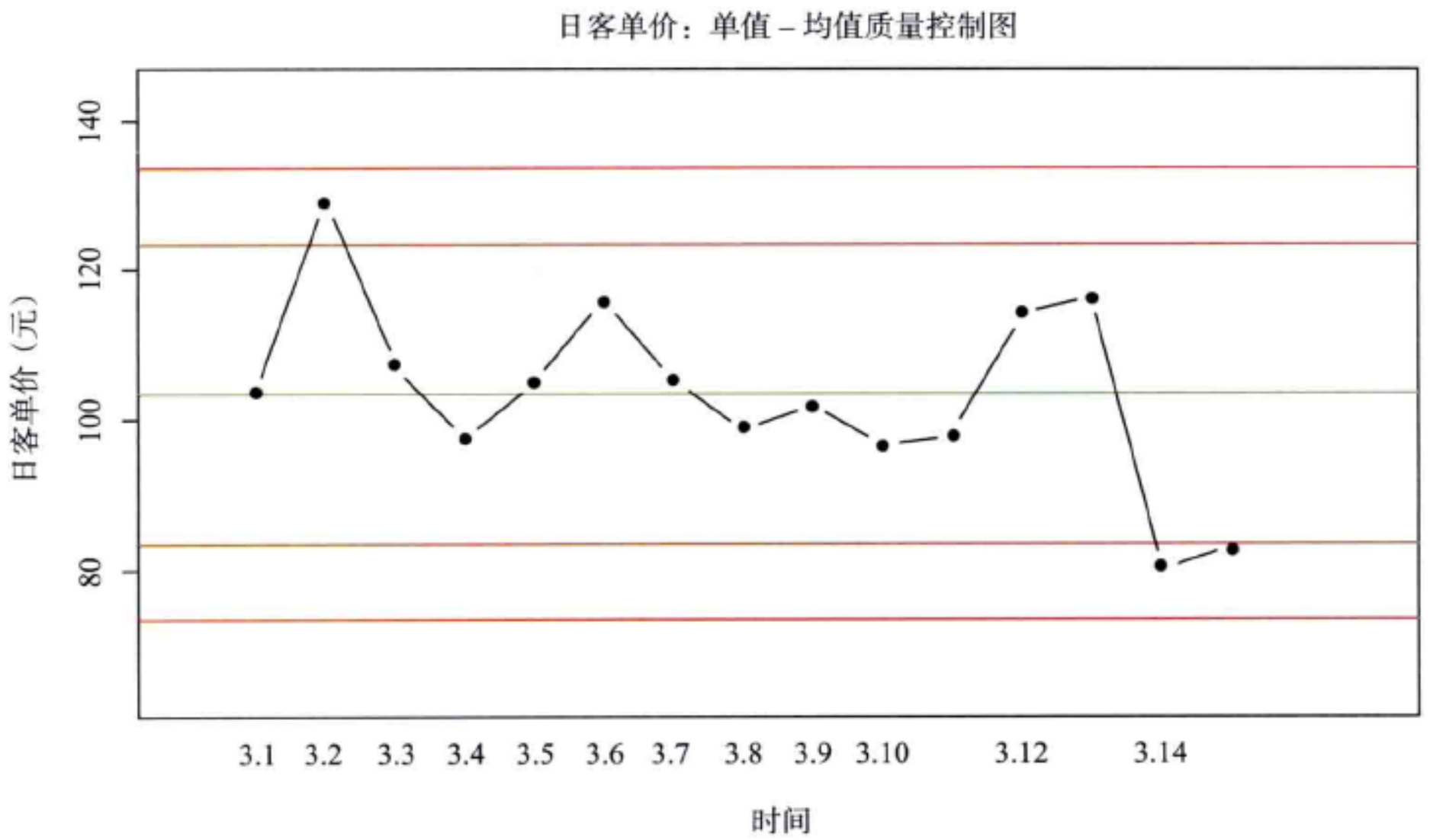


图 4-20　客单价指标的单值均值控制图

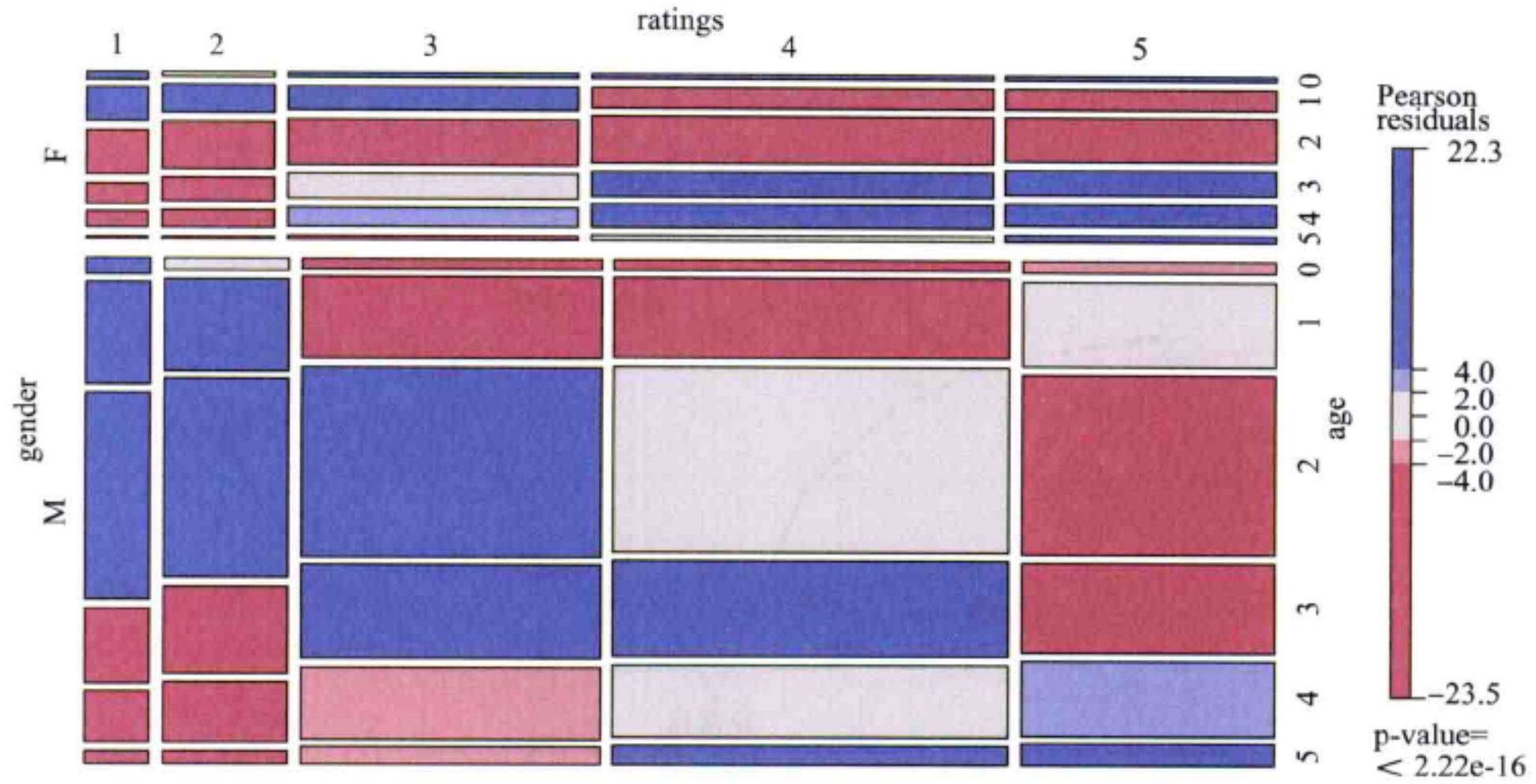


图 4-33　结果展示

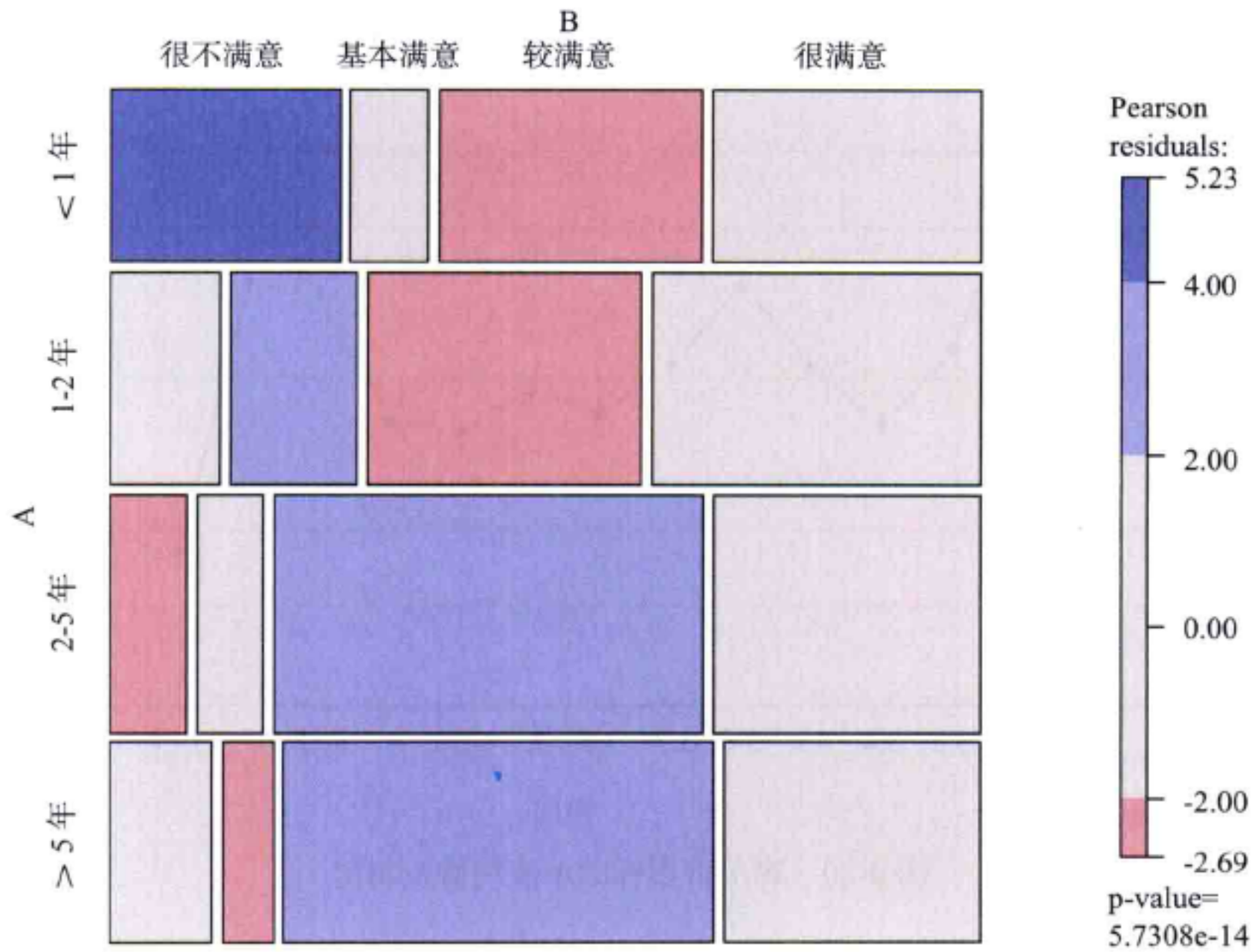


图 4-35　结果显示

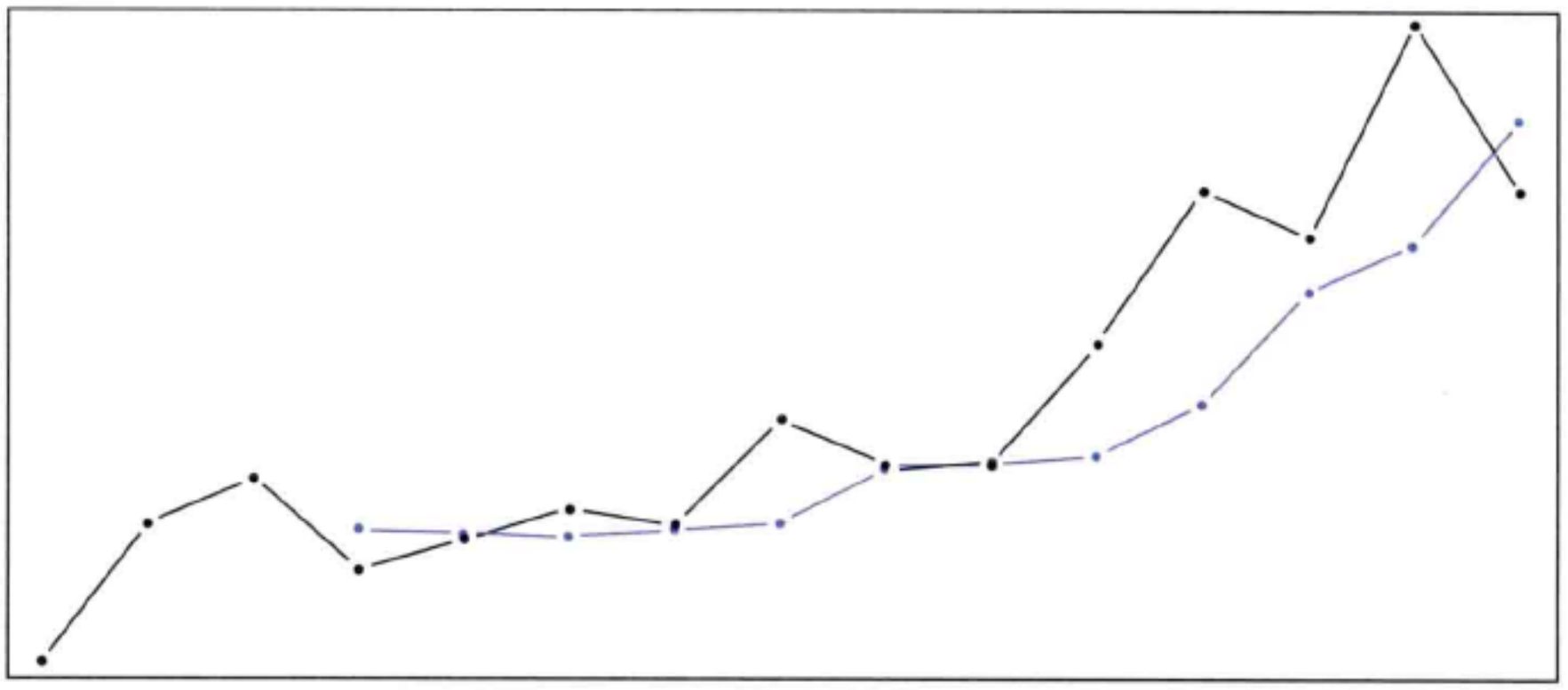


图 5-6　原始数据和加权移动平均处理后的数据对比图

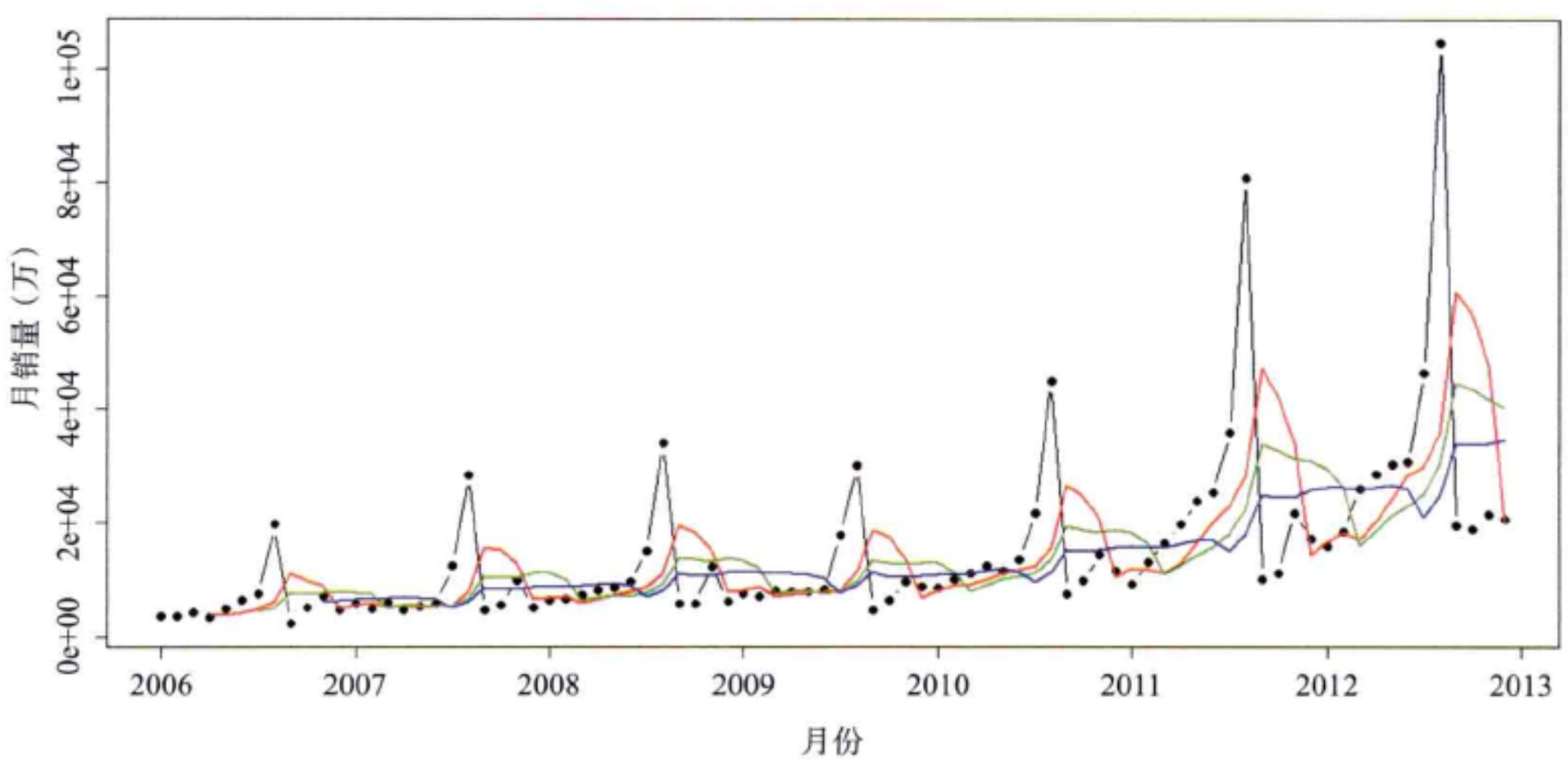


图 5-7　月销量数据和移动平均处理效果图

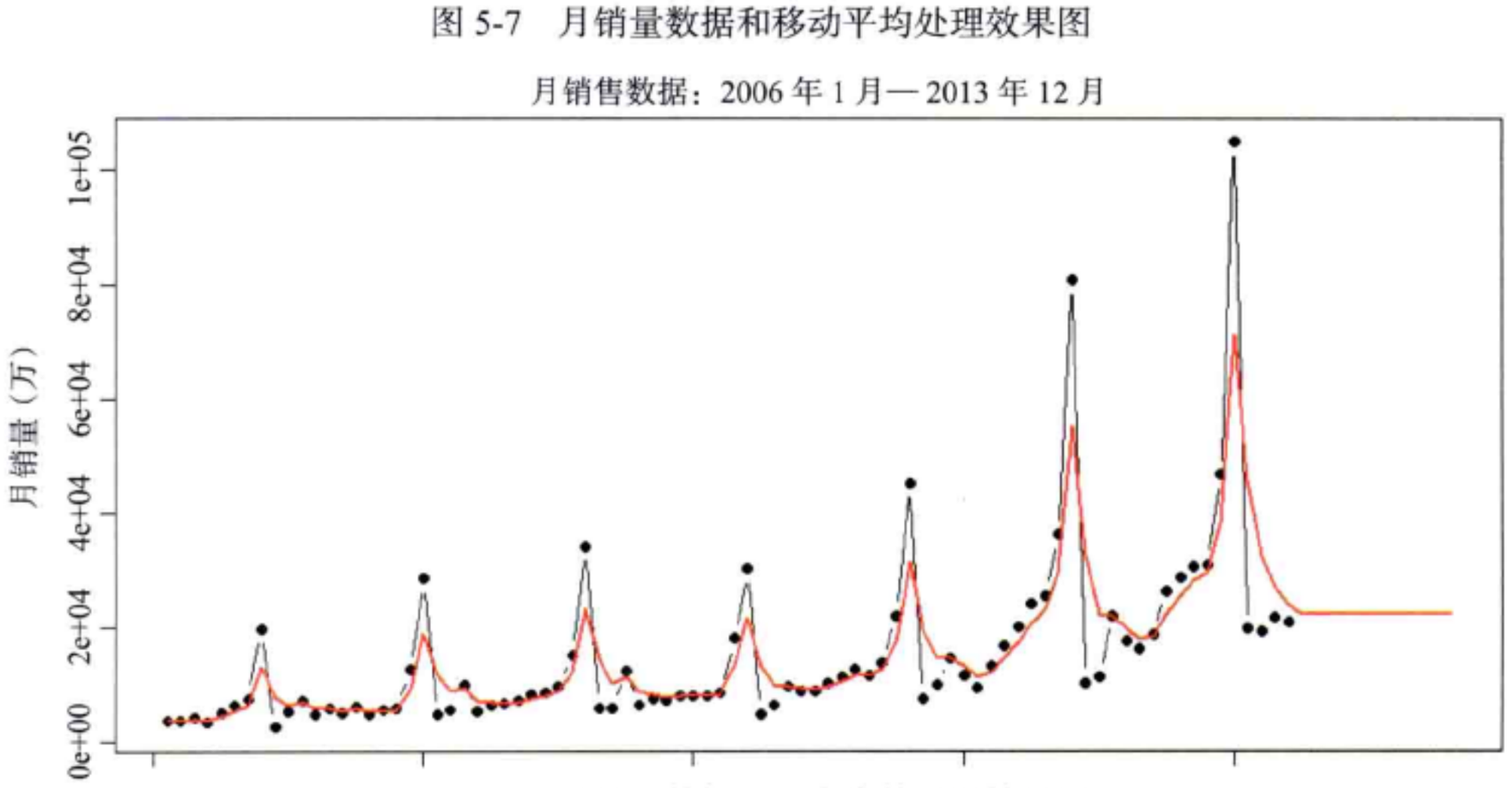


图 5-8　月销量数据和一次指数平滑效果图

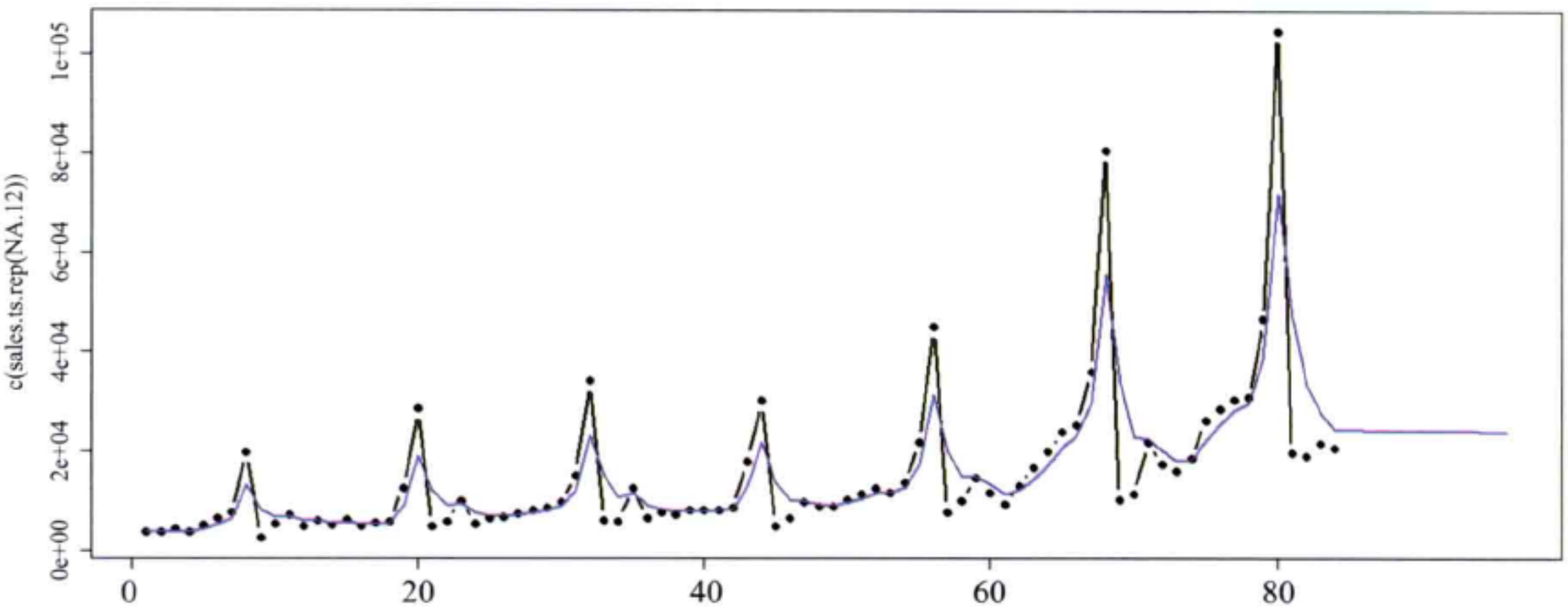


图 5-12　月销量数据和二次指数平滑效果图

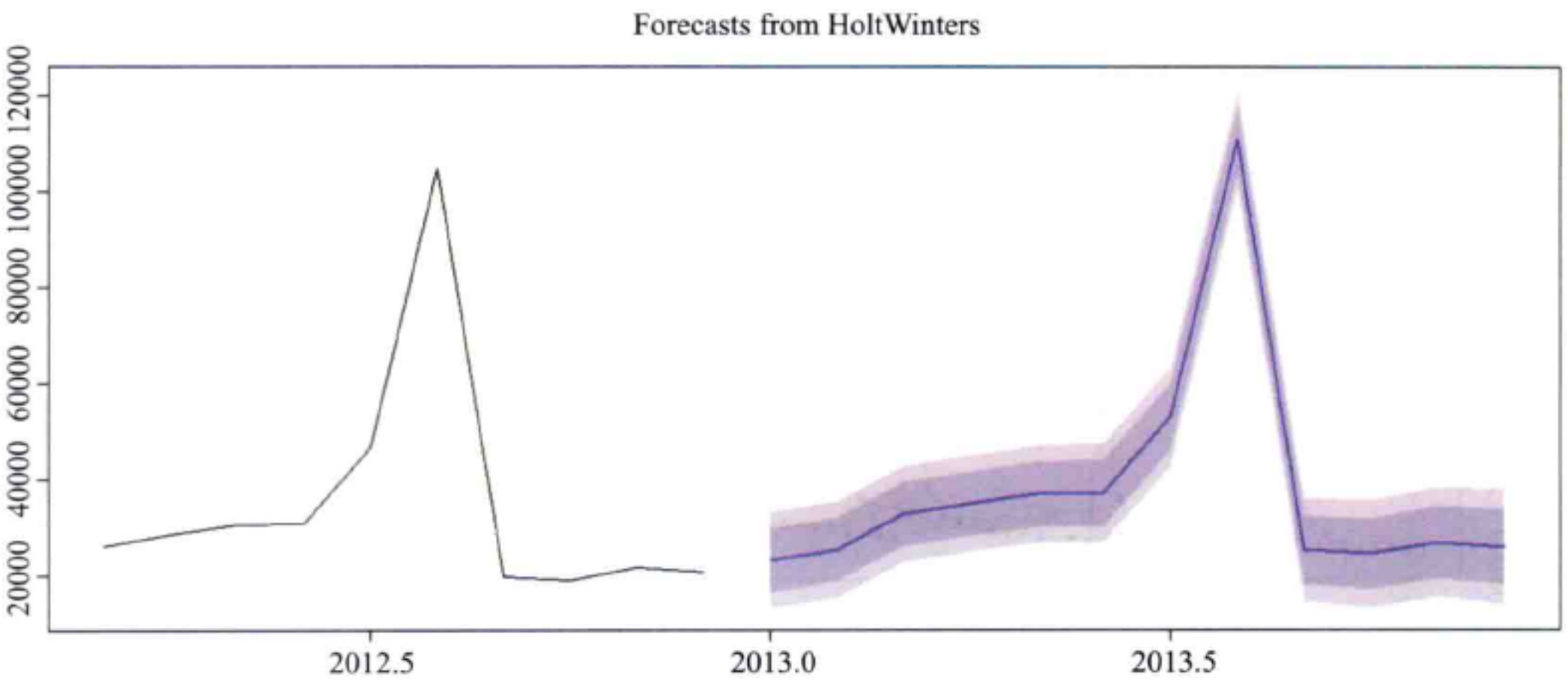


图 5-14　三次指数平滑的预测及区间图

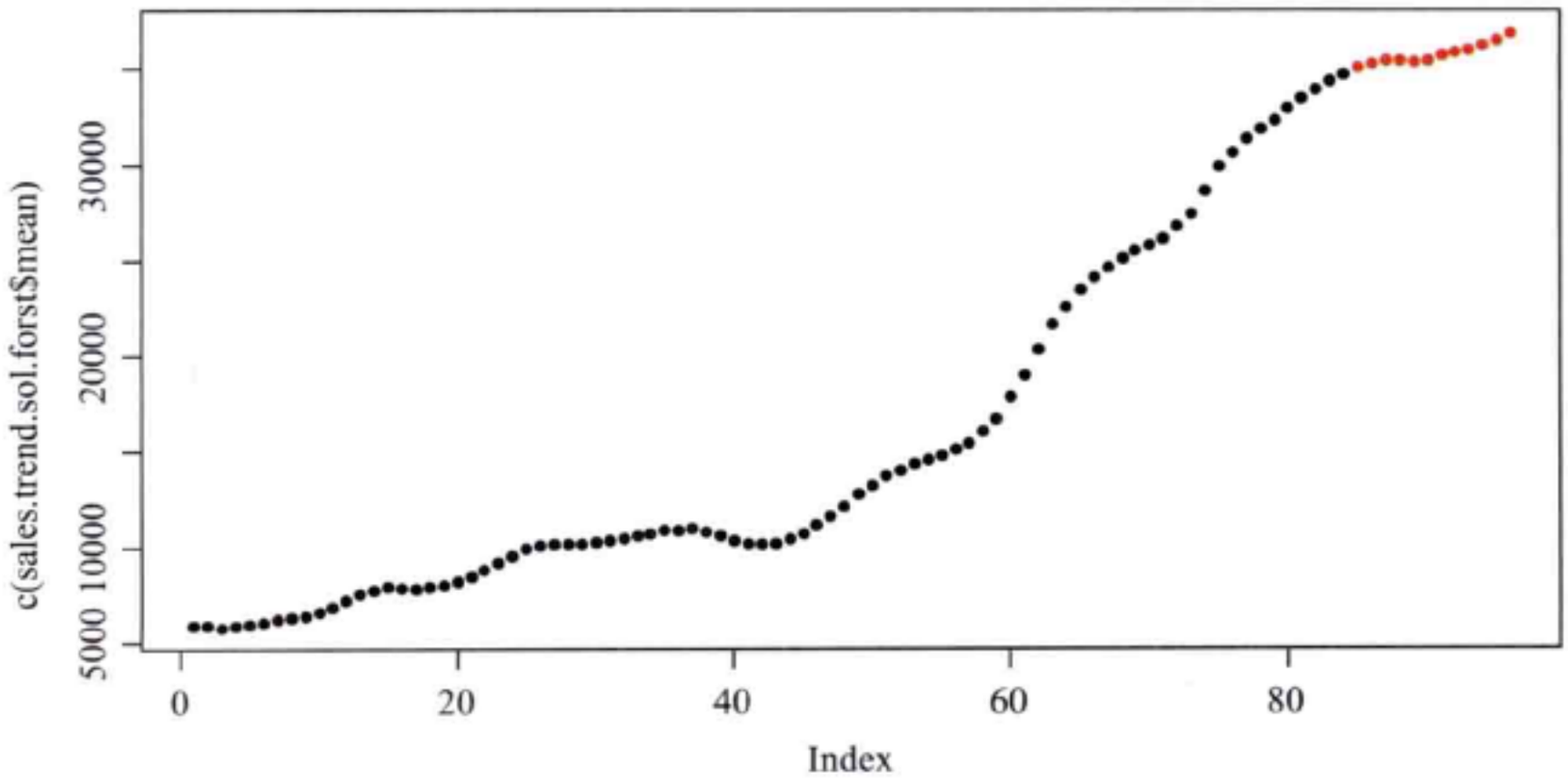


图 5-32　月销售数据的趋势预测图

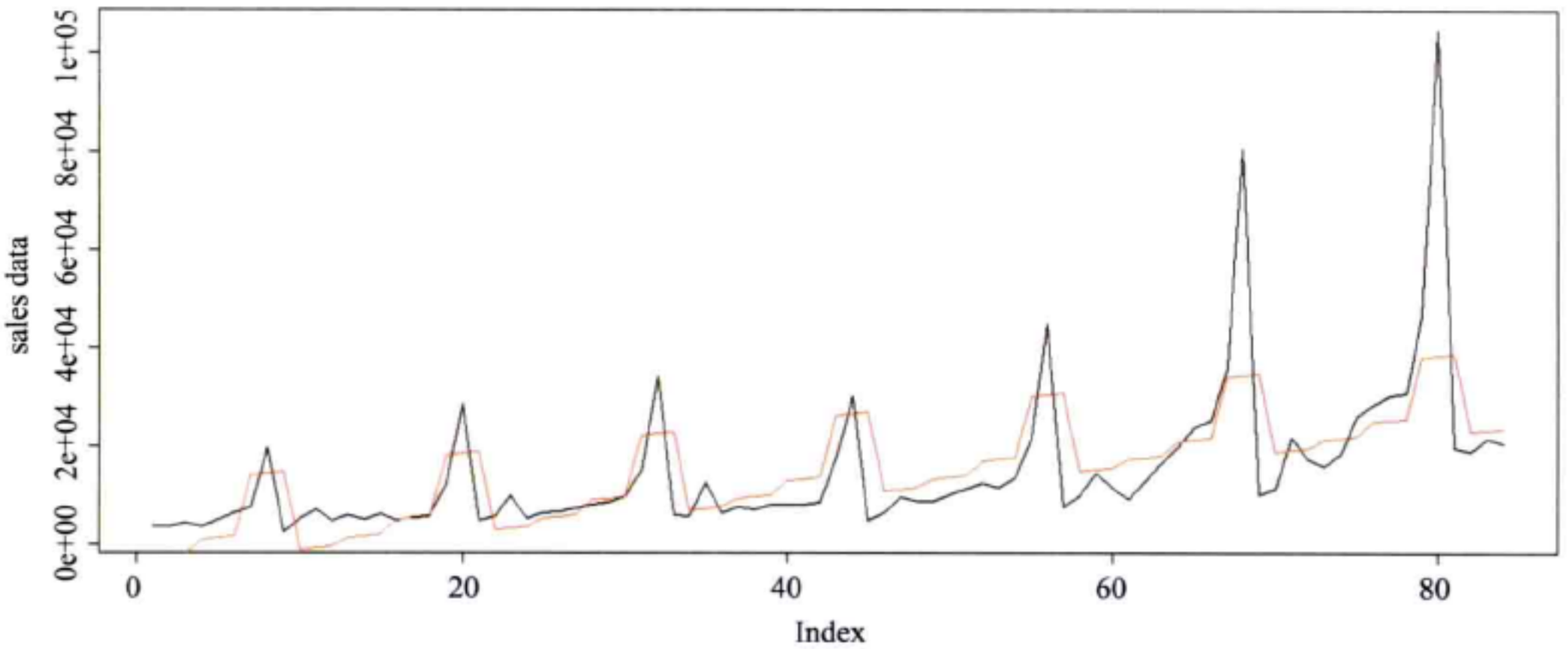


图 6-9　原有数据和模型预测数据的曲线

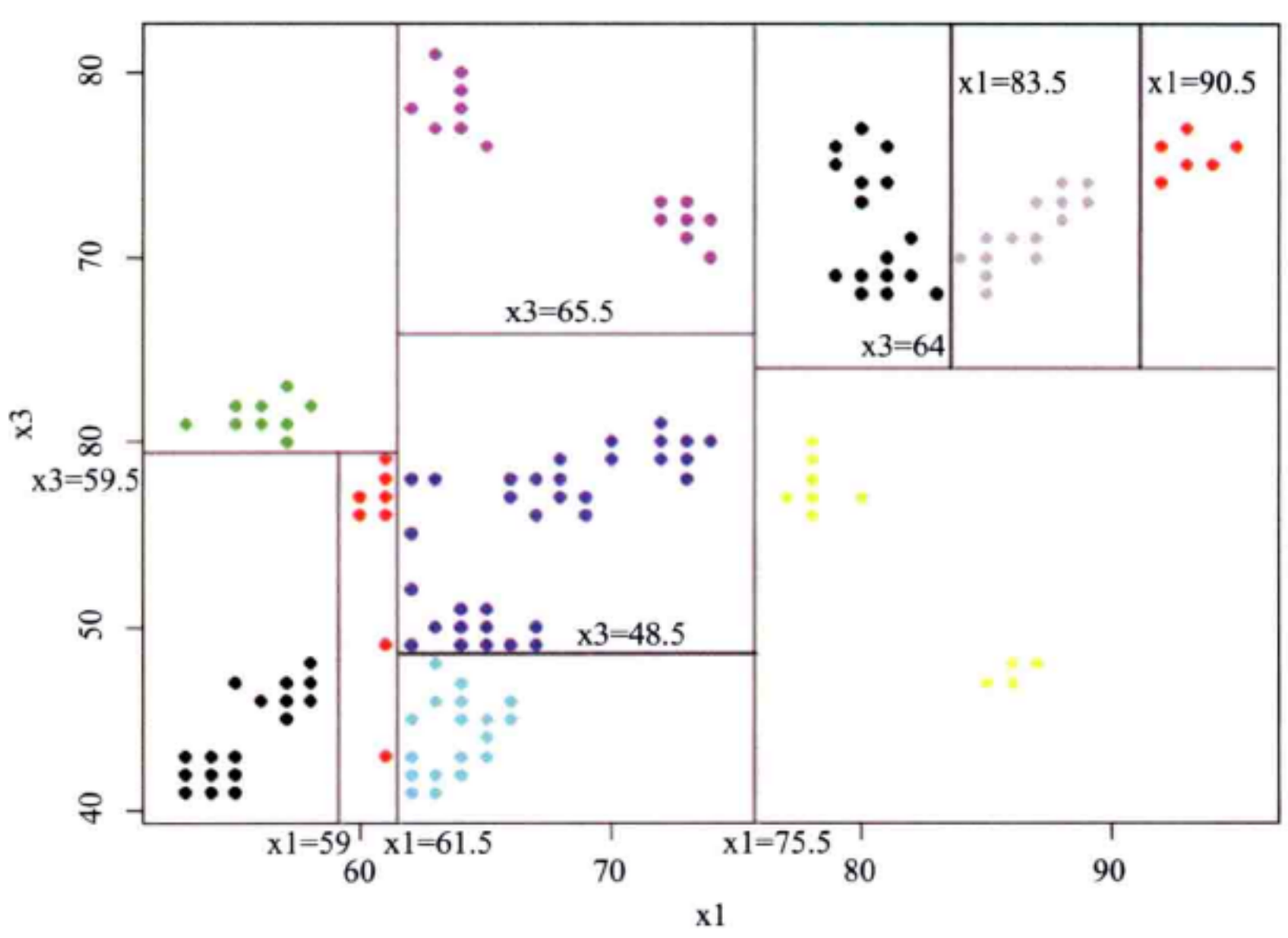


图 6-15　散点分割图

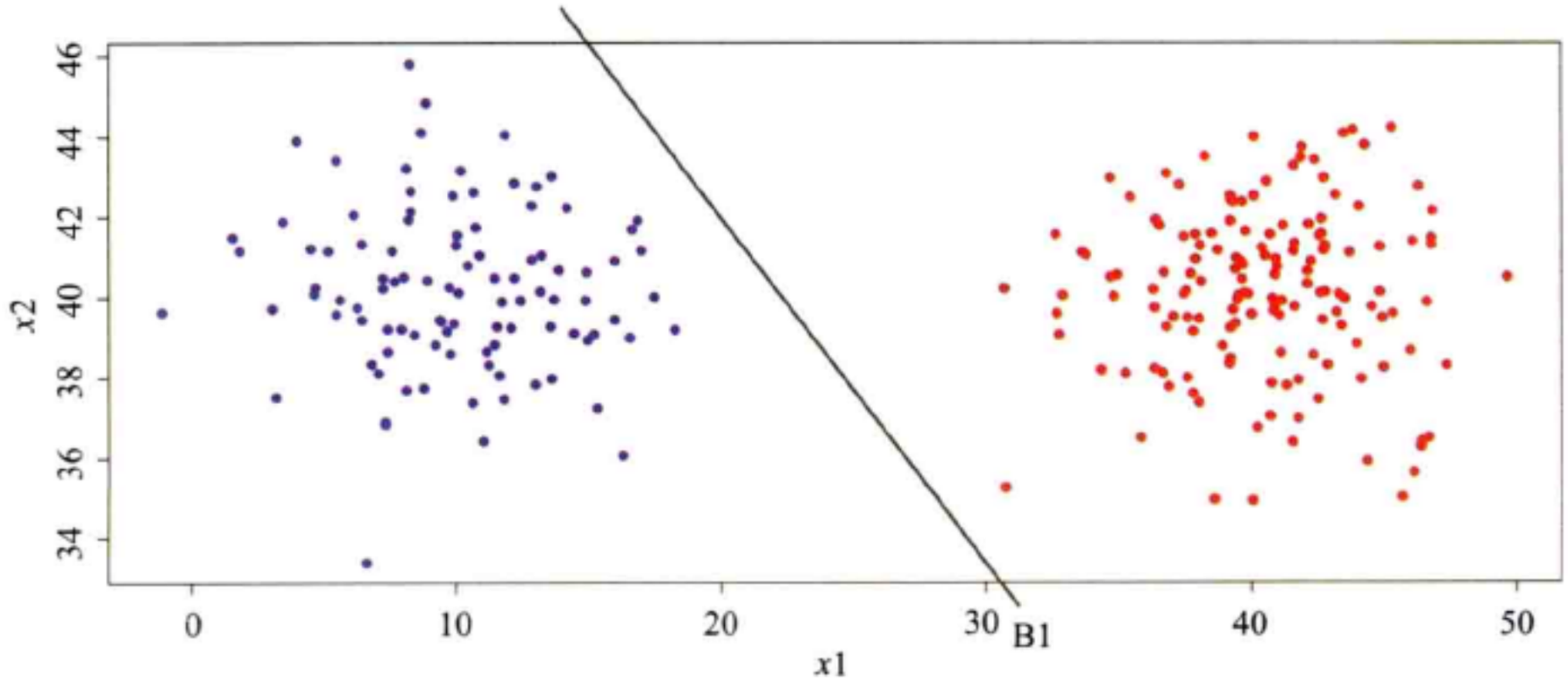


图 7-15　示例图

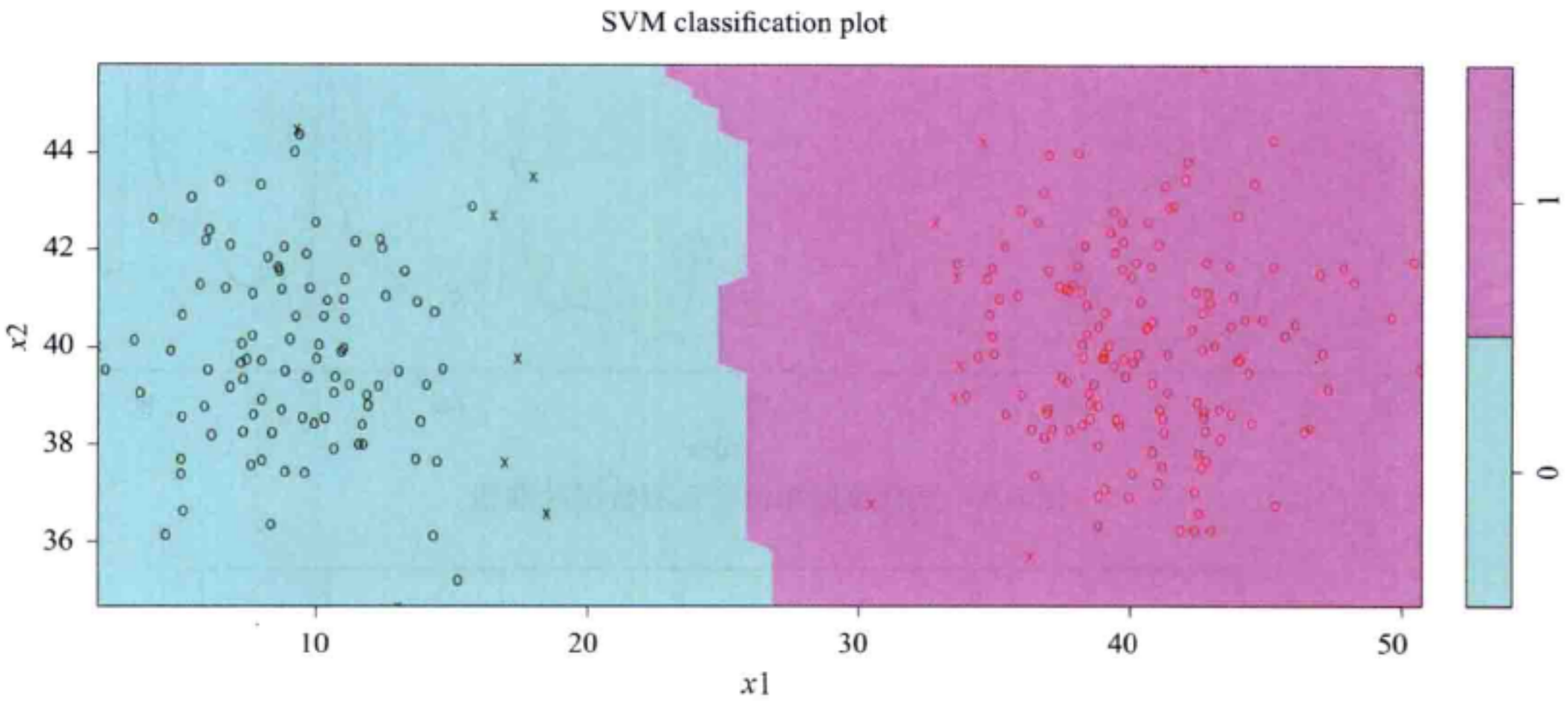


图 7-18　结果展示图

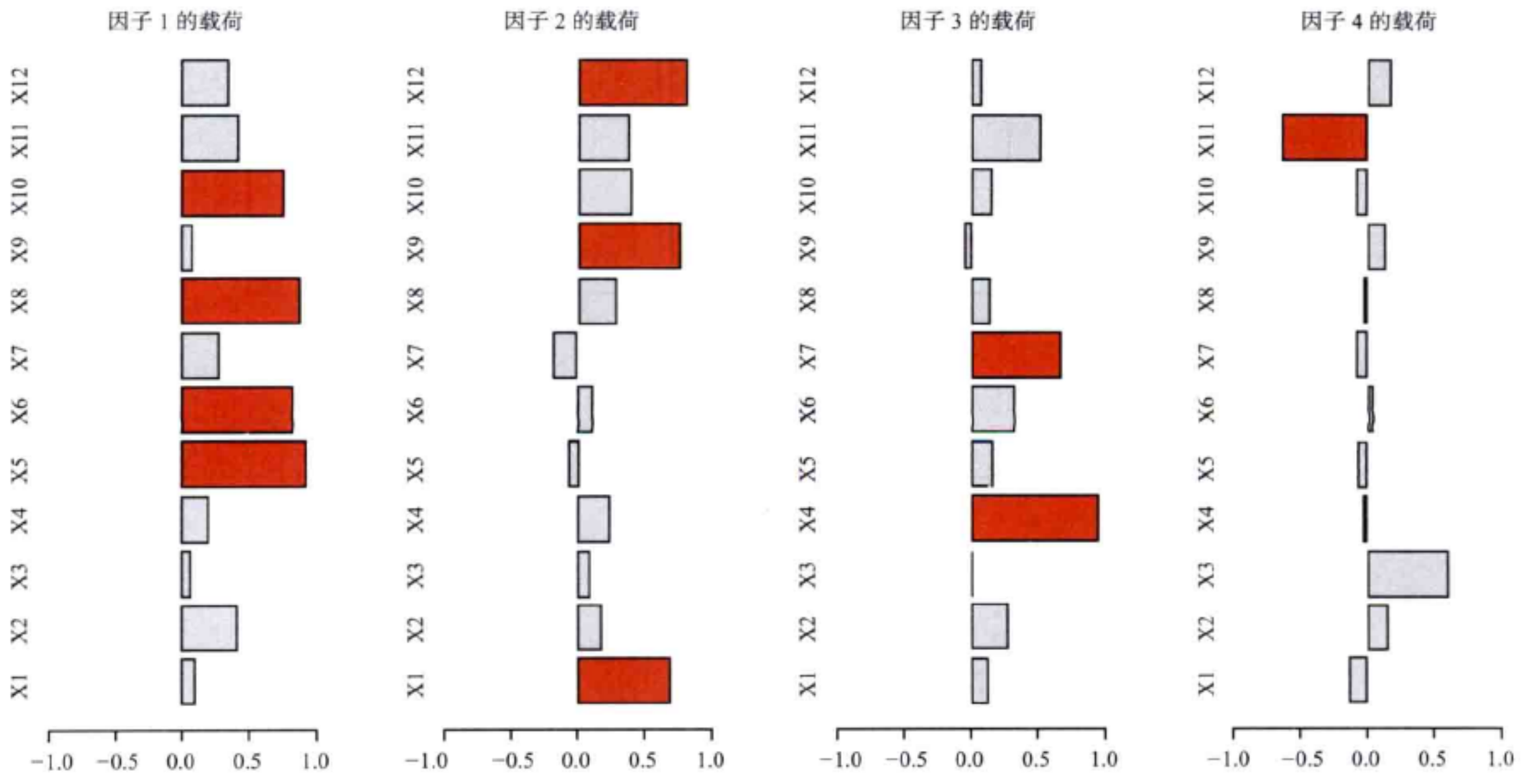


图 8-1　因子载荷示意图

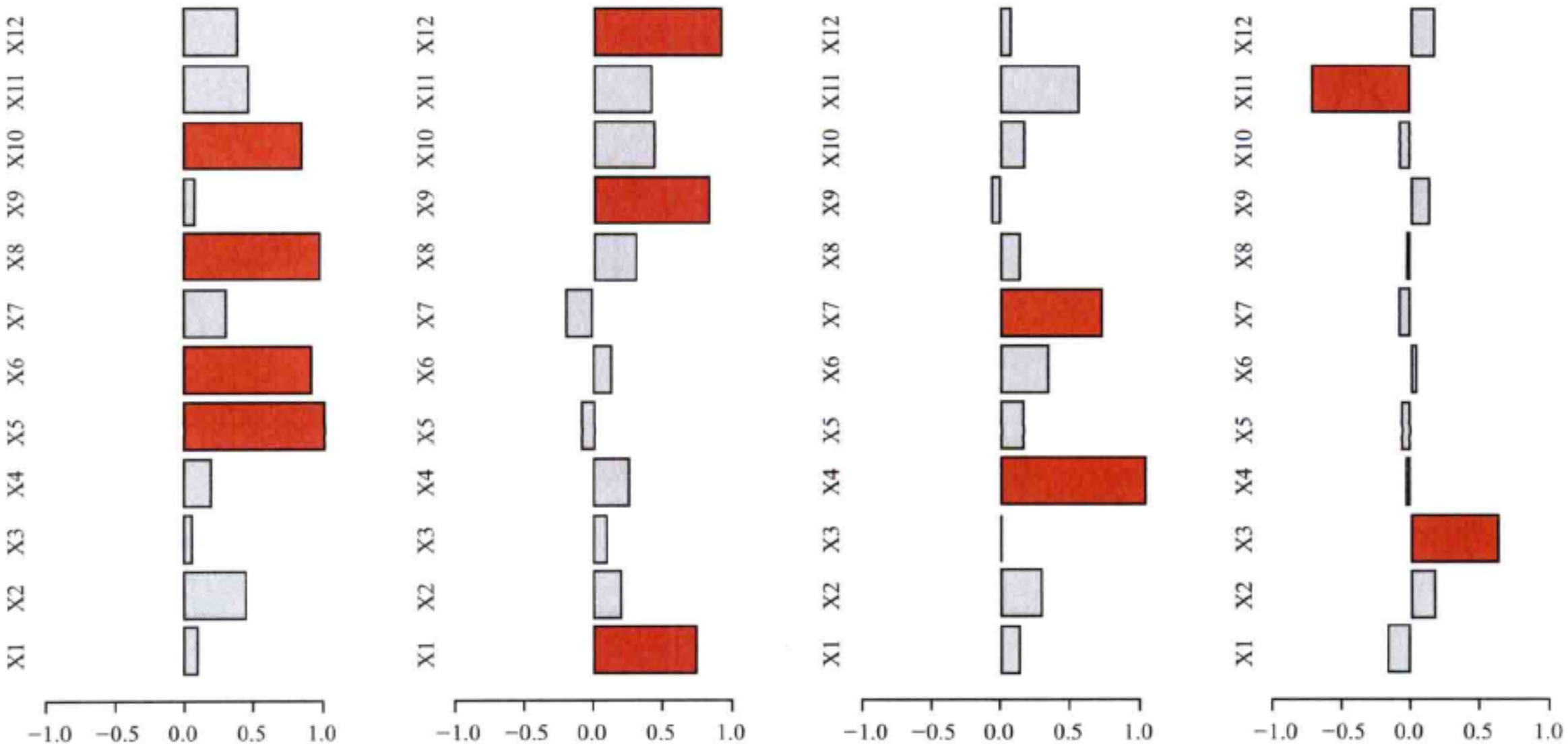


图 8-3　因子载荷结果

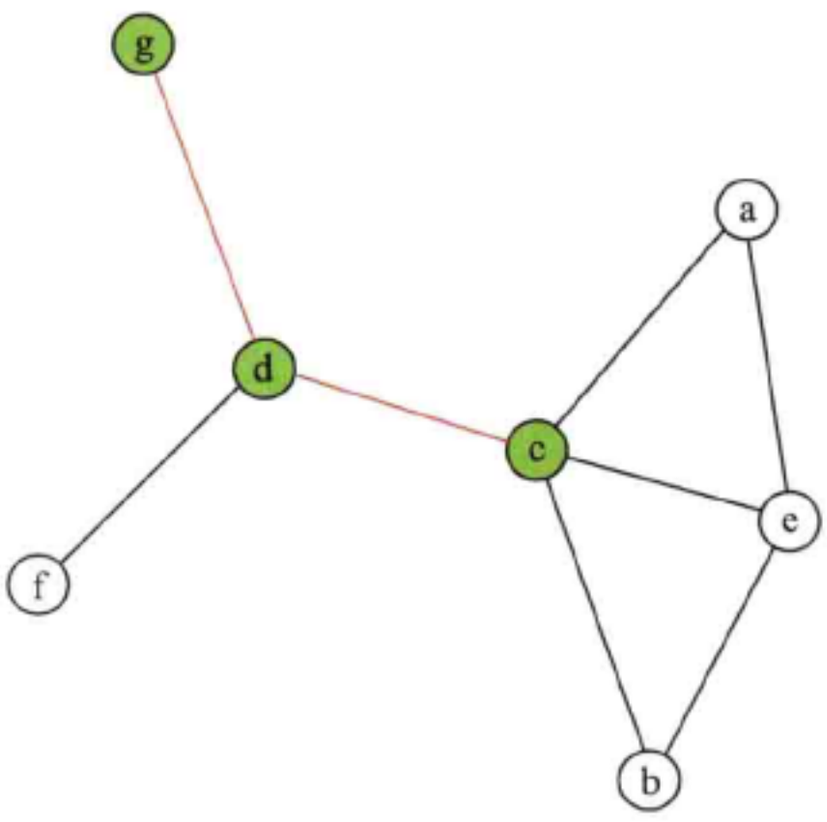


图 9-12　结果展示图

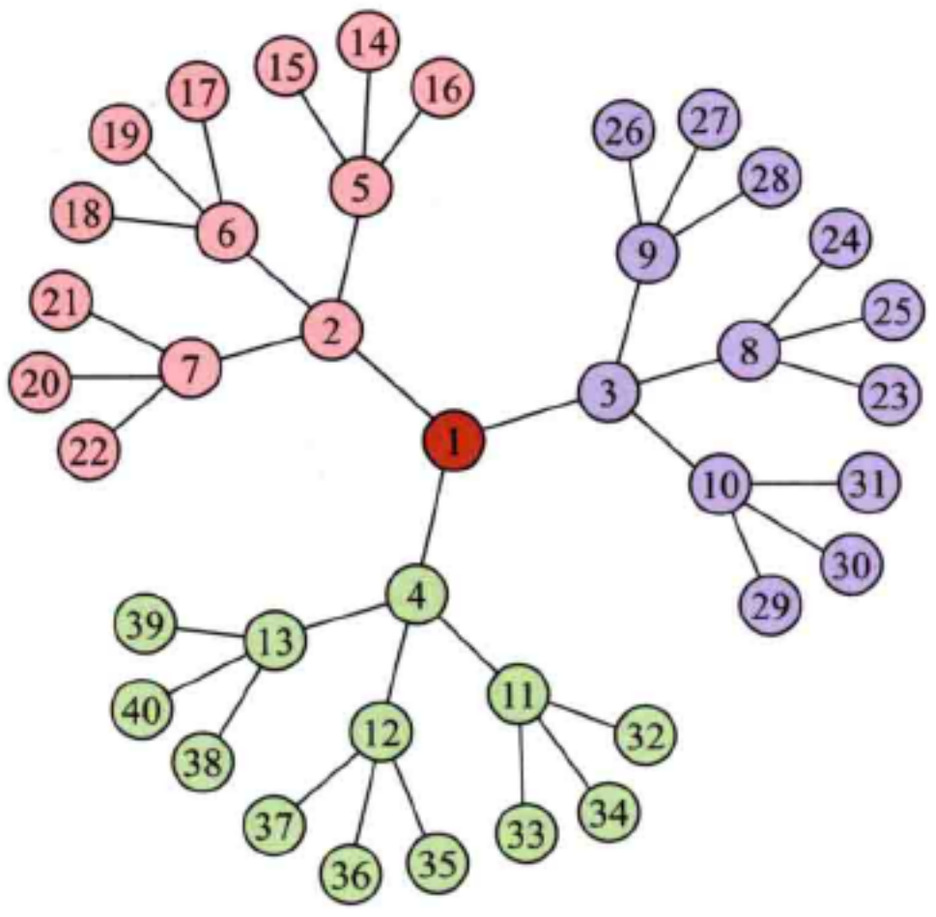


图 9-14　示例图

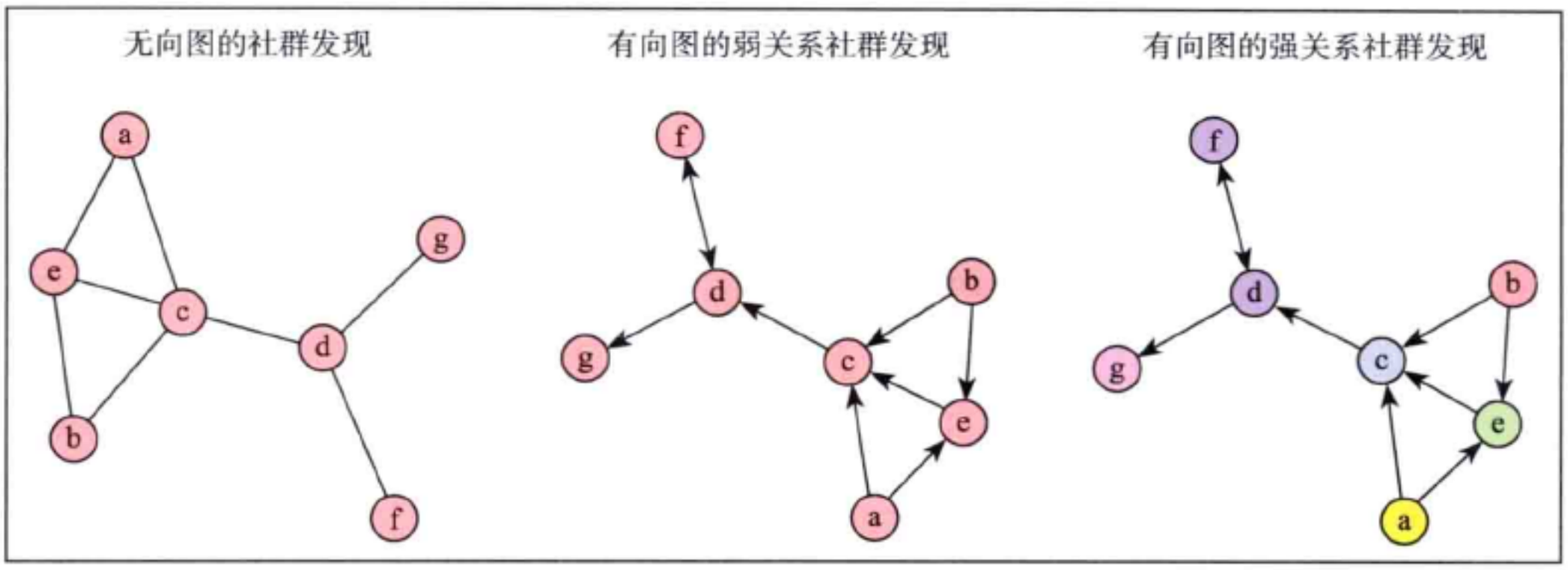


图 9-17　基于点连接的社群发现示例图

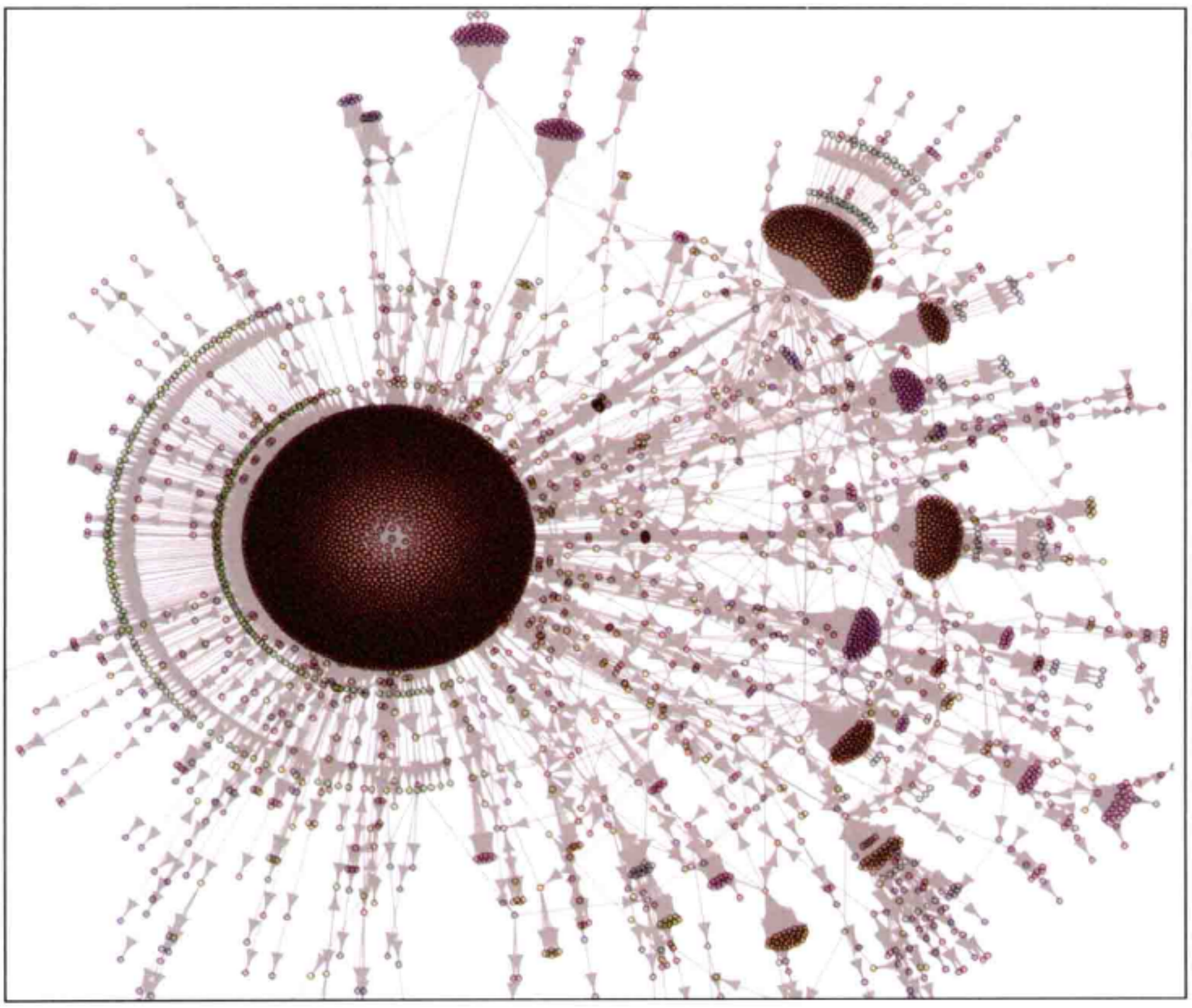


图 9-20　微博传播网络图

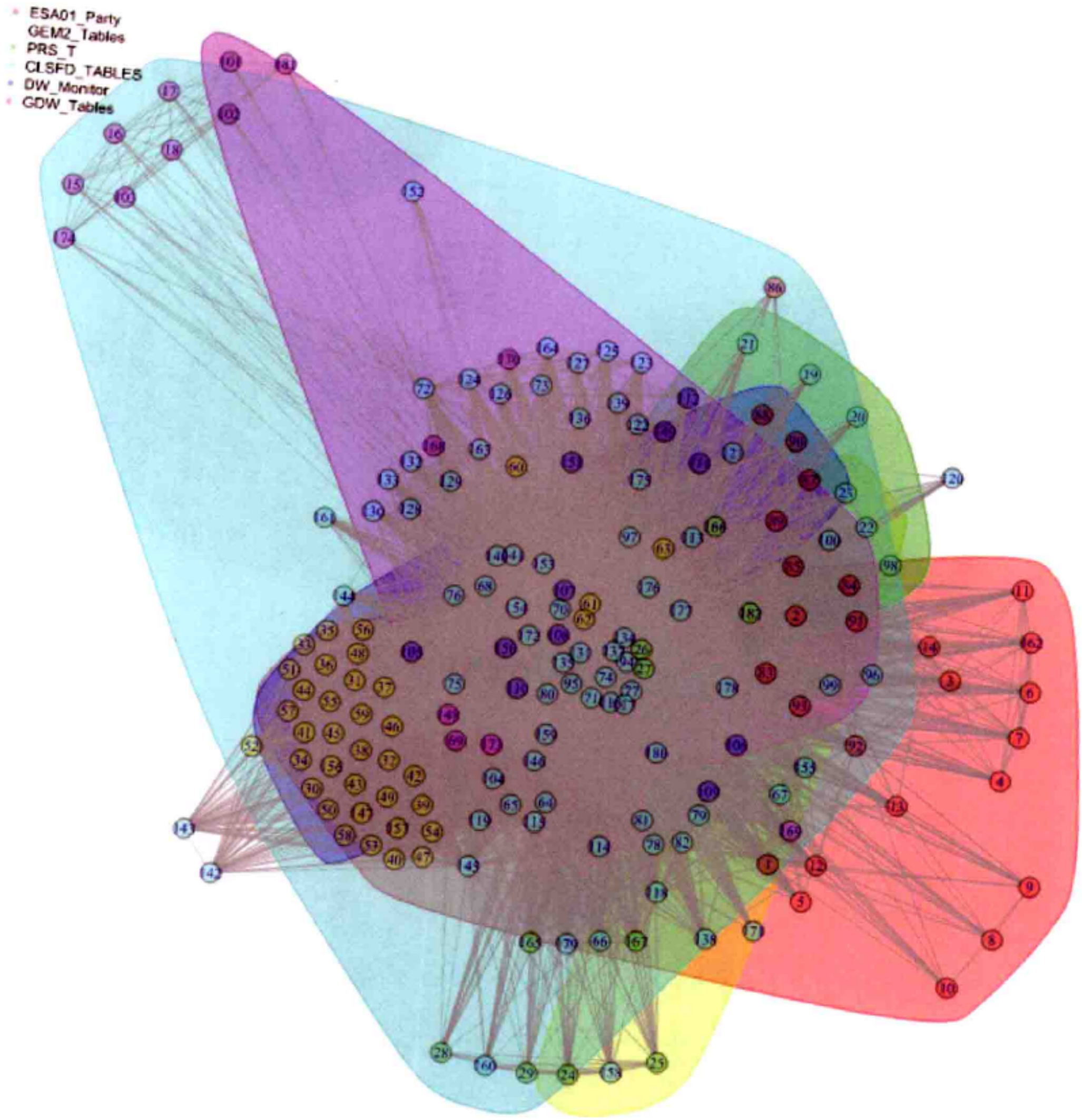

ESA01_Party
GEM2_Tables
PRS_T
CLSFD_TABLES
DW_Monitor
GDW_Tables

图 9-25　结果展示

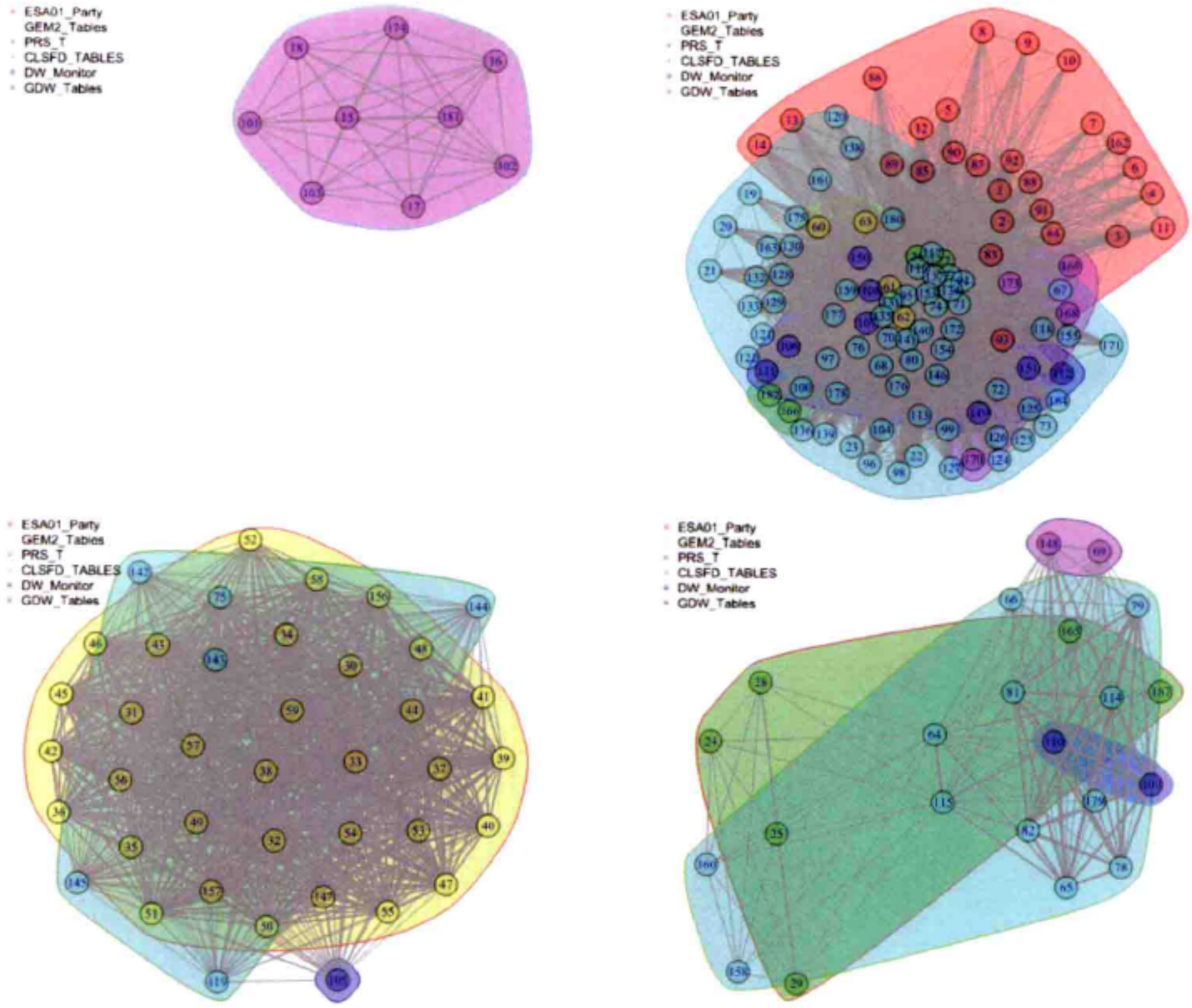


图 9-26　结果展示

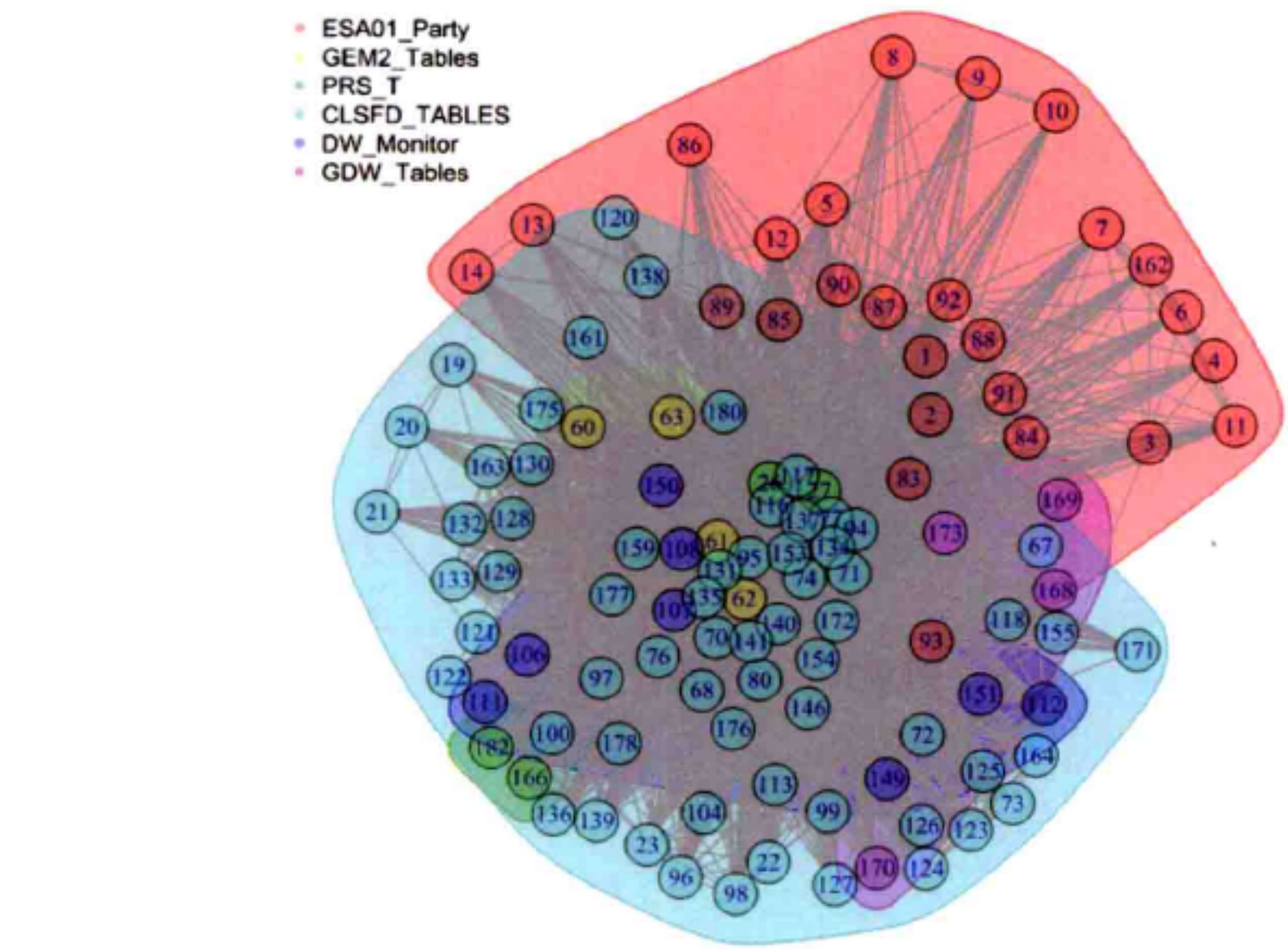


图 9-28　社群 2 展示图

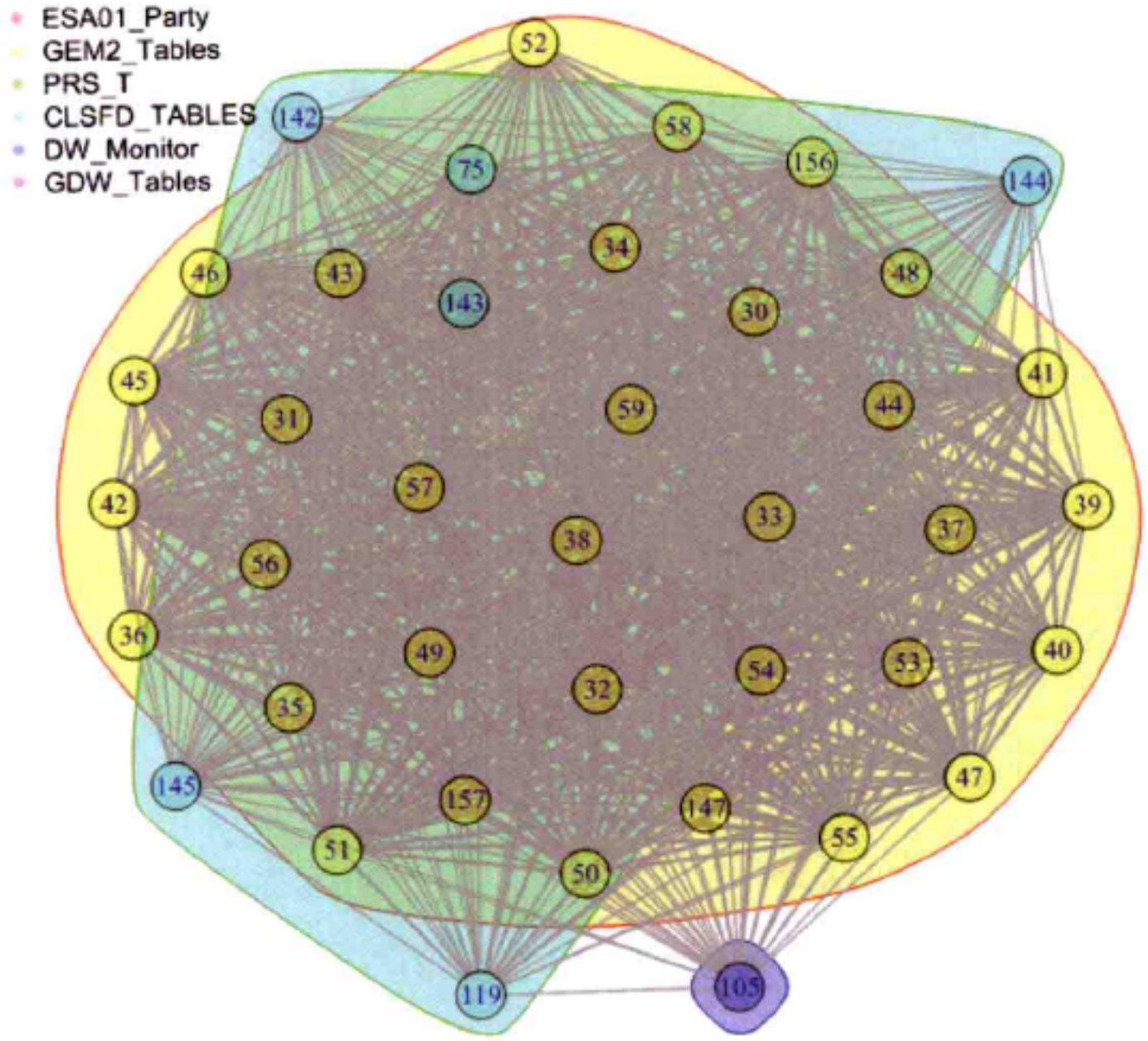


图 9-29　社群 3 展示图

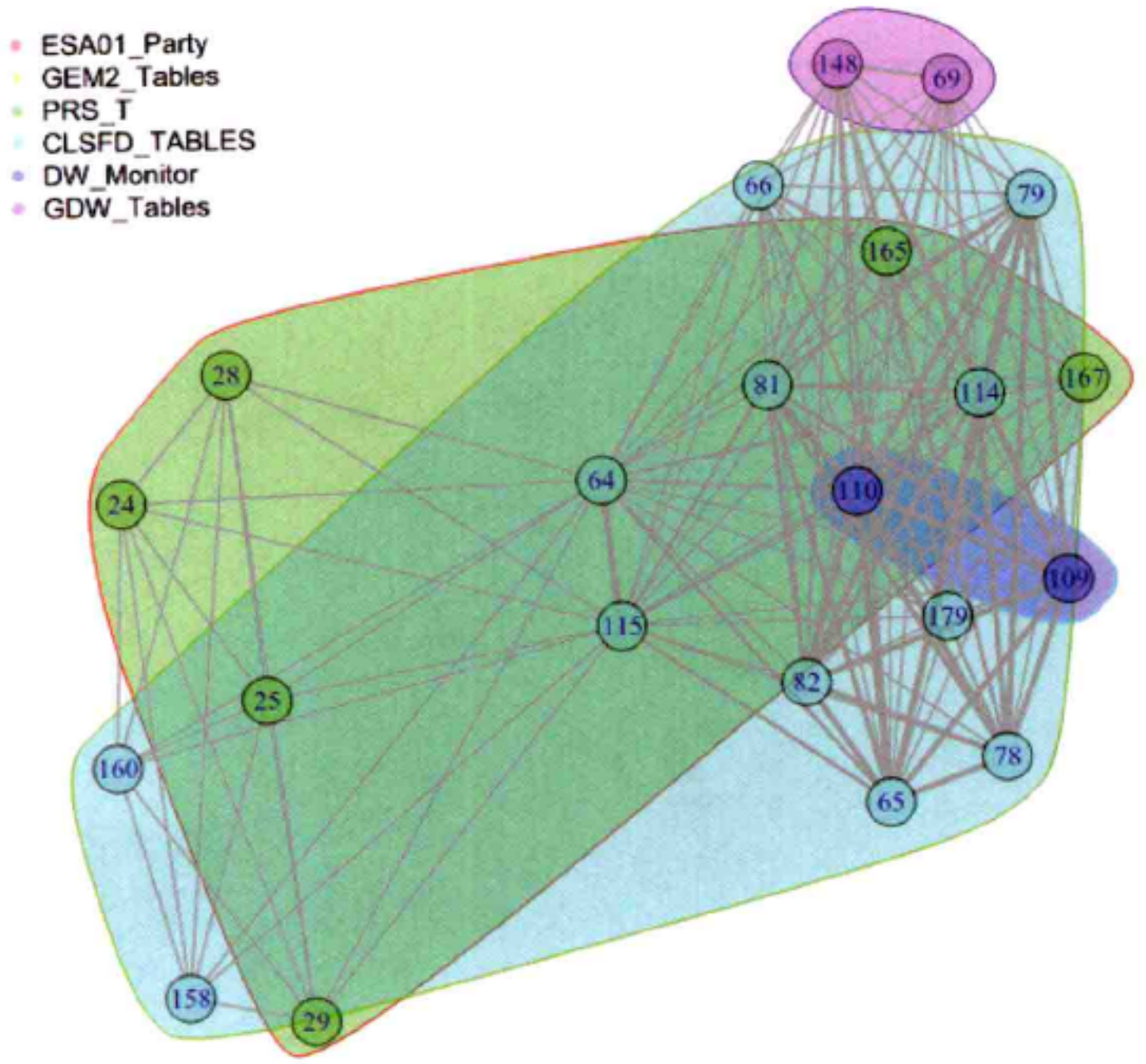


图 9-30　社群 4 展示图

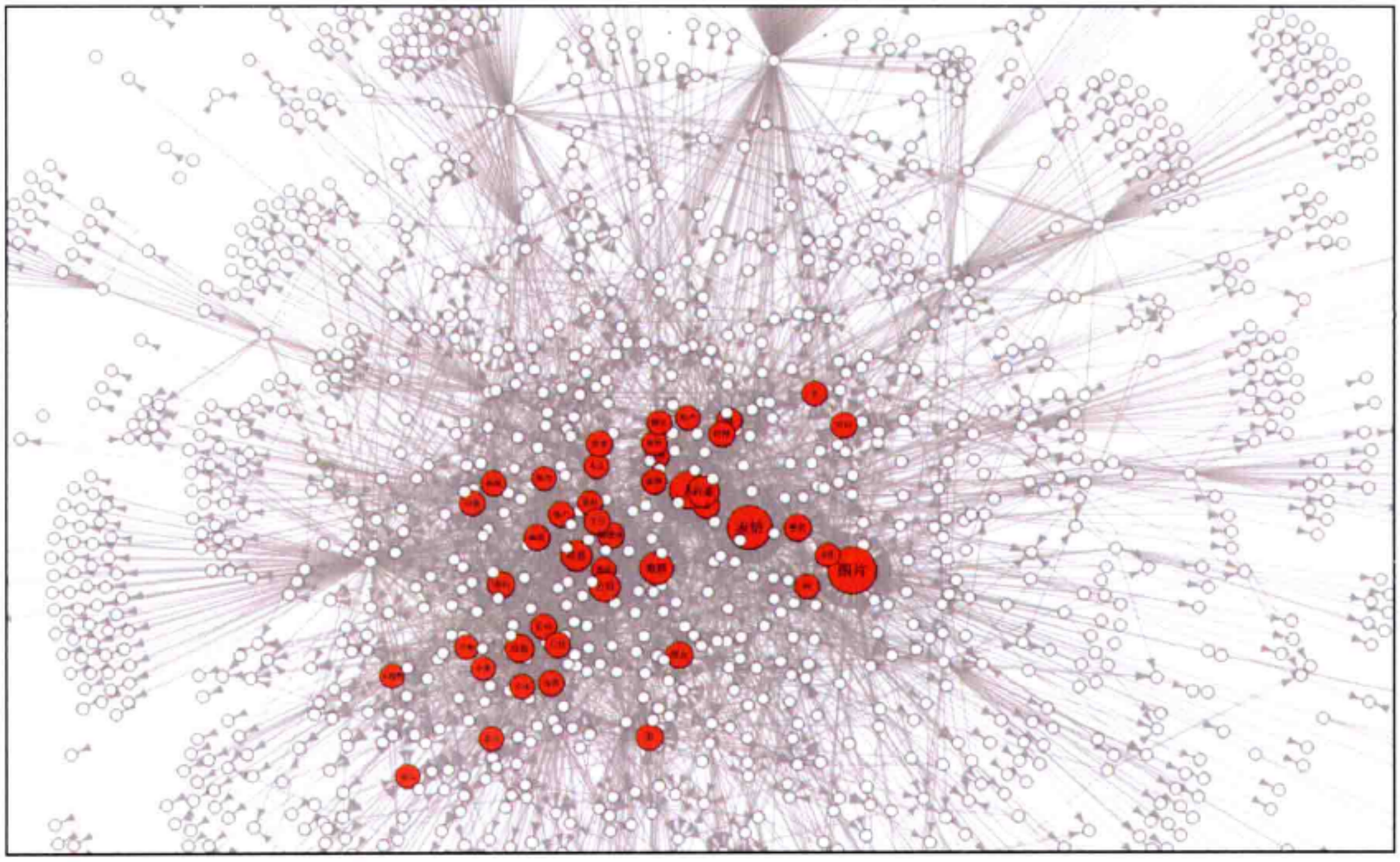


图 10-21　重点词条网络图

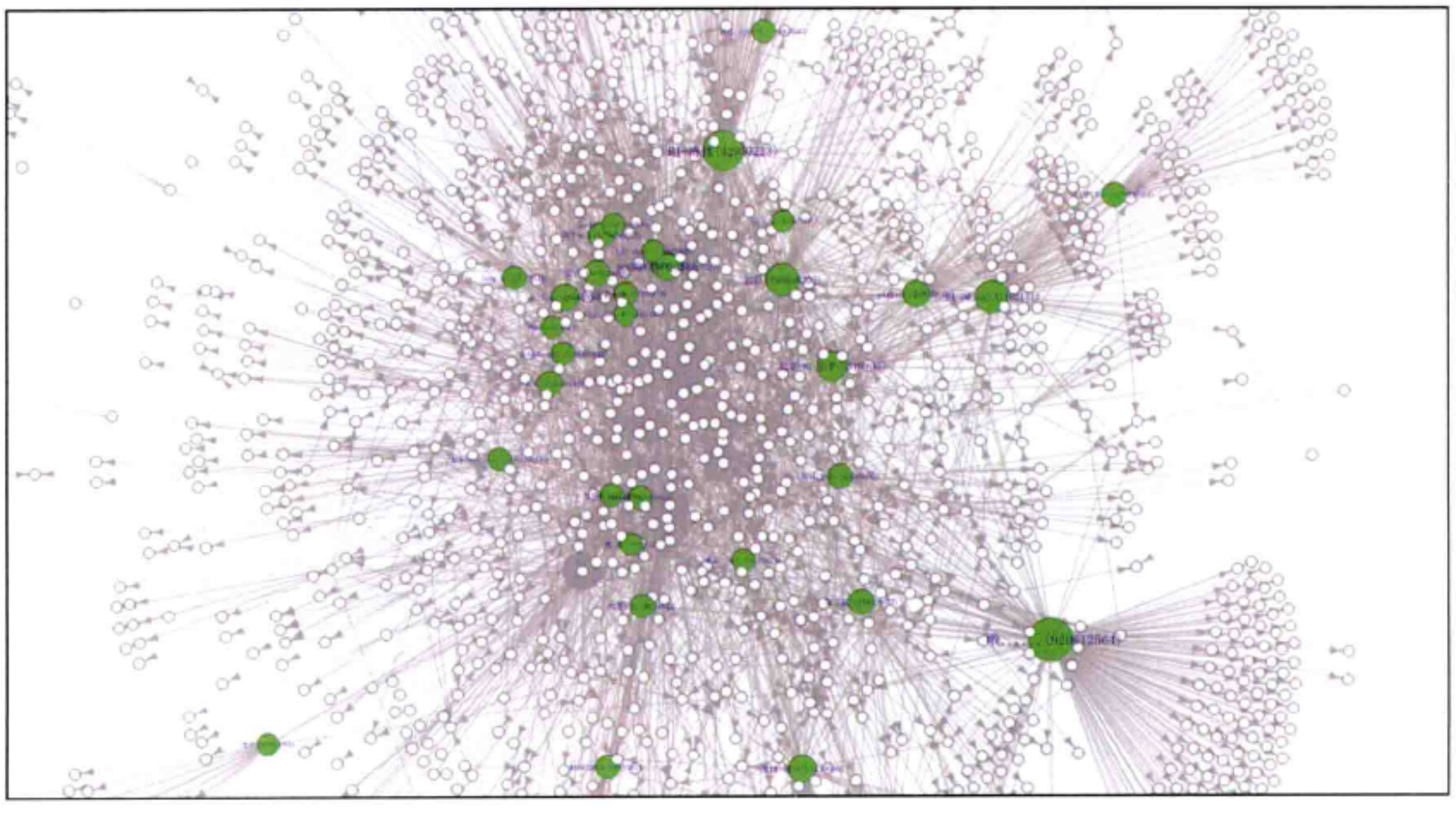


图 10-22　重点用户网络图

前　言

大数据下网站分析的重要性

当前，随着互联网对人们生活影响的不断增大，每天有数以亿计的用户行为数据被网站记录。这些数据以隐性信息的方式反映人们的行为习惯、兴趣爱好、性格特征，甚至是政治主张。网民每天都在告诉我们，他们到底需要什么。而这也正反映了用户的需求，可以说谁能读出用户的需求，谁就能更好地迎合用户，获取更多的流量，其网站的用户体验也就越好。

网站分析就是通过用户使用网站时留下的数据来读出用户需求、习惯、爱好的方法。同时，通过数据分析也可以驱动整个网站的优化。利用数据分析驱动网站优化是一种最科学的决策方式，它直接反映了在现实中使用网站的用户情况，而非网站产品经理或决策人想象的用户。

R 语言的优势

R 语言作为一个可进行交互式数据分析和探索的强大平台，被越来越多的数据分析和挖掘人员及公司所使用。其优势主要体现在以下几个方面：

- R 语言的使用和学习并不需要掌握高深的语法，这使得分析师可以把更多的精力放在算法本身的实现上。
- R 语言是目前包含数据算法最全的工具，其灵活强大的绘图功能可以更好地表现数据处理结果。
- R 语言本身是免费和开源的，并与众多商业数据分析软件，如 SAS 等有接口。
- R 语言有搭建可视化软件的扩展包，公司可以依据自身的商业需求搭建指标监控平台。

R 语言的优势使其发展势头大好，特别是在互联网公司，R 语言逐渐受到了重视。对于有志成为互联网数据挖掘师 / 分析师的读者来说，掌握 R 语言将成为他们未来必备的技能之一。

编写本书的主旨

如何才能将 R 语言和网站分析有机地结合起来，使其价值得到体现？对于广大读者来说，很需要一本对此进行详细阐述，深入分析，指导其进行实战的图书，但是目前市面上还没有相关的图书出版，而本书的主旨正是为了填补这个空缺。相信本书会给读者带来不小的收获，并且可以把 R 语言更好地应用到网站分析和互联网数据挖掘的工作上。

本书侧重于 R 语言对网站分析和挖掘的代码实现，并不侧重于对计算结果的分析讨论以及网站统计的思维方法。同时，本书以通俗的非学术语言结合实际的代码案例来介绍网站分析的基本思路和统计理论的数学原理，力争做到对读者授之以“渔”，而非简单地授之以“鱼”。

阅读对象

- 网站数据分析师。通过阅读本书，结合公司自身的业务逻辑，使用 R 语言进行网站数据系统的搭建。
- 数据挖掘师。书中结合互联网的常用数据挖掘方法，给出 R 语言的相应实现方式，使原来以 SAS/SPSS 软件为基础的挖掘方案转化为以 R 软件为基础的挖掘方案。
- 互联网产品经理。过去网站分析系统是以自主研发或使用第三方免费工具的方式来实现的，希望通过本书可以实现以 R 语言为基础的新的搭建方案。
- 其他行业的数据从业人员。虽然本书与网站分析和互联网数据挖掘相结合，但里面涉及的经典分析算法是同其他行业相通的，所以对于其他行业的数据分析和挖掘也有借鉴作用。

校勘和支持

由于作者的水平有限，编写的时间也很仓促，书中难免会出现一些错误或者不准确的地方，恳请读者批评指正。读者可以在我的个人博客 www.bassary.com 中留言，或直接发送至我的邮箱 568677413@qq.com。我会定期发布 Bug 勘误表，同时，如果你遇到任何问题，也可以通过上述途径来提问，我将尽量在线上为读者提供最满意的解答。书中的全部代码都是经过本人调试的（R 语言版本为 2.9.2），部分源文件会发布在我的博客中，我也会将相应的功能更新及时更正出来。

感谢

首先要感谢 R 语言的创建者 Ross Ihaka 和 Robert Gentleman，他们开创了一款影响统计学乃至所有自然和社会学科的软件。

感谢沈阳理工大学，它为我大学四年提供了一个轻松而充实的学习环境。感谢我的大学导师马平全，是他培养了我如何自学和探索新学科的能力。

感谢一直活跃在群里（238649264）和关注我博客的网友：王思远、秦赫赫、徐小小、费正飞、王洋等，他们为本书的形成提出了很多建设性的意见。

感谢机械工业出版社华章公司的编辑杨绣国，在这一年多的时间里始终支持我的写作，你的鼓励和帮助引导我顺利完成全部书稿。

感谢我的老婆刘慧，是她一直鼓励我直到完成全书。这里我想对她说："能有你的陪伴是我一生的幸运。"

最后一定要感谢我的母亲和已故去的父亲，感谢你们将我培养成人，并时时刻刻给我信心和力量!

谨以此书献给我最亲爱的家人和朋友，以及众多热爱 R 语言和从事网站分析挖掘工作的朋友们。

李明

推荐阅读



资深数据分析师多年一线工作和培训经验结晶，20余位行业专家和企业管理者联袂推荐！

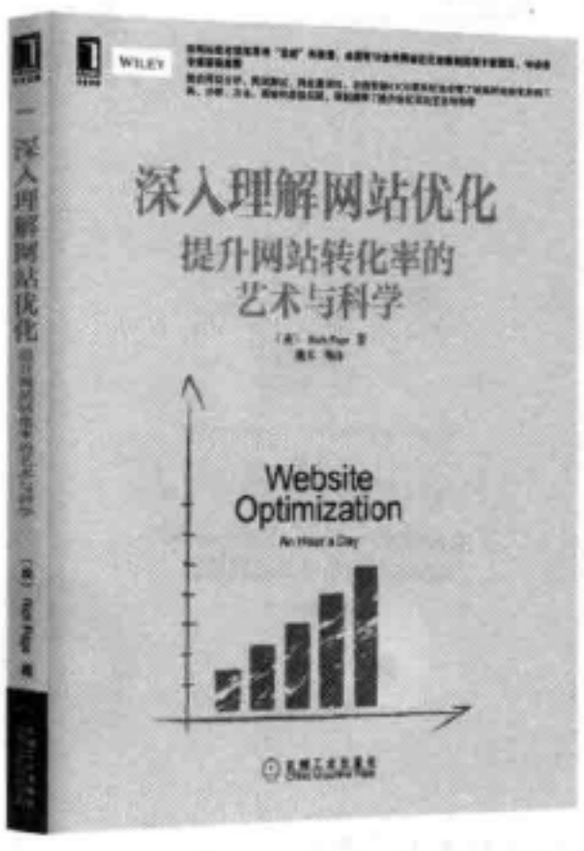
数据分析领域开创性著作，内容生动、有趣，而又不失专业和严谨

以实际的企业经营案例为依托，复杂的数据分析知识巧妙地融入互动式情景中，系统呈现数据分析的专业知识、方法和思路，以及数据分析项目的完整流程



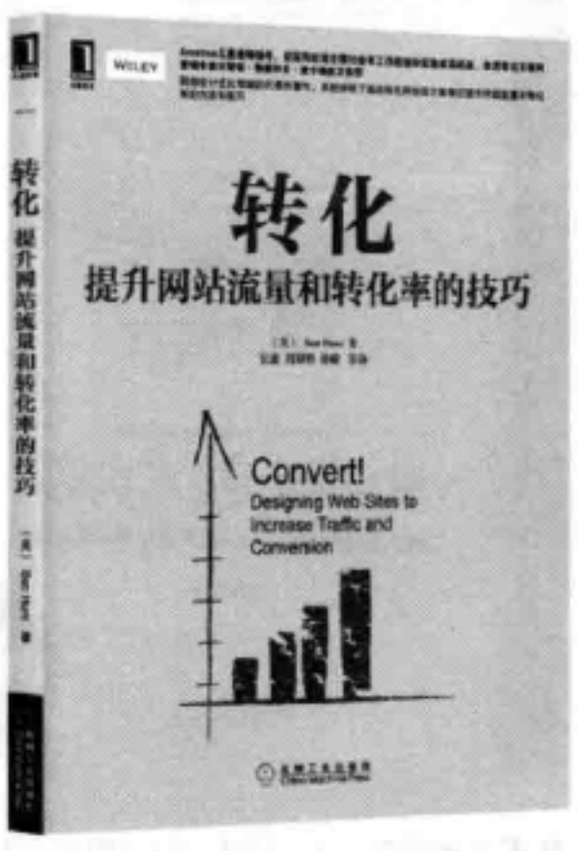
网站分析领域的开创性著作，首度将六西格玛管理工具和思想引入网站分析全流程，使分析思维和方法更科学化和系统化

全面讲解网站分析必备的各项知识和技术，充分讲解Excel在网站分析中的各种功能和用法，包含大量案例和最佳实践



在网站优化领域享有“圣经”的美誉，由拥有10余年网站优化经验的资深专家撰写，10余位专家联袂推荐

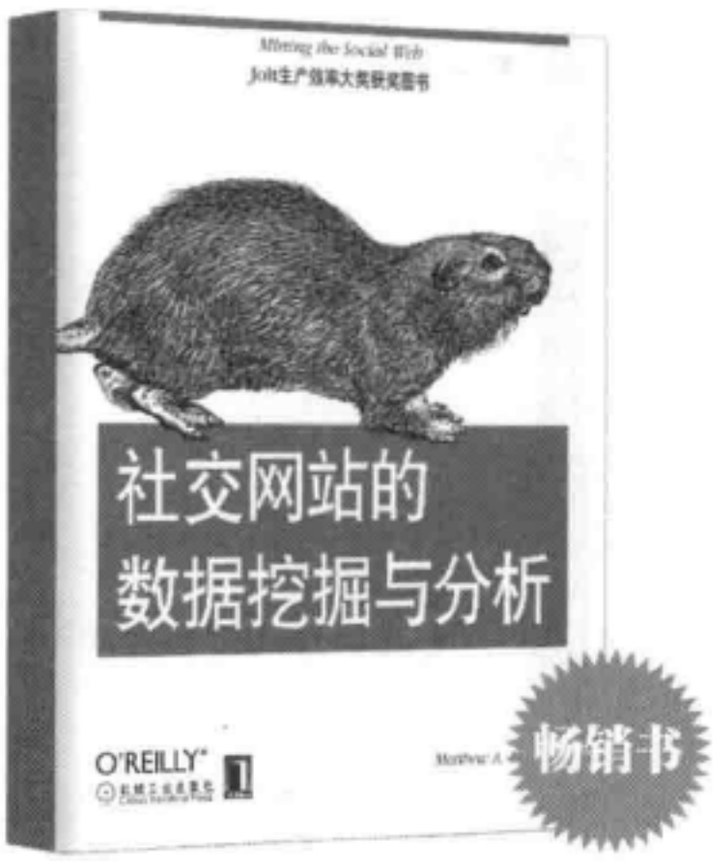
结合网站分析、网站测试、网站易用性、在线营销4大方面系统地讲解了提高网站转化率的工具、步骤、方法、策略和最佳实践，深刻阐释了转化率优化的艺术与科学



Amazon五星级畅销书，资深网站设计师20余年工作经验和实验成果结晶，世界著名互联网营销专家杜雷顿·勃德和肯·麦卡锡鼎力推荐

网站设计优化领域的代表性著作，系统讲解了通过优化网站设计来有效提升网站流量和转化率的方法和技巧

推荐阅读



第21届Jolt生产效率大奖获奖图书

社交数据挖掘、分析必备参考书

不仅对初学者有用，对数据挖掘的资深人士也极具参考价值。



第1版广受好评，第2版基于Hadoop及其相关技术最新版本撰写，从多角度做了全面的修订和补充。

不仅详细讲解了新一代的Hadoop技术，而且全面介绍了Hive、HBase、Mahout、Pig、ZooKeeper、Avro、Chukwa等重要技术，是系统学习Hadoop技术的首选之作



资深Hadoop技术专家撰写，EasyHadoop和51CTO等专业技术社区联袂推荐！

从源代码角度深入分析MapReduce的设计理念，以及RPC框架、客户端、JobTracker、TaskTracker和Task等运行时环境的架构设计与实现原理。

深入探讨Hadoop性能优化、多用户作业调度器、安全机制、下一代MapReduce框架等高级主题。



腾讯资深Hadoop技术专家撰写，EasyHadoop和51CTO等专业技术社区联袂推荐！

从源代码角度深入分析Common和HDFS的架构设计与实现原理，为Hadoop的优化、定制和扩展提供原理性指导。

从源代码中参透分布式技术精髓与分布式系统设计的优秀思想和方法。

目　　录

前言

第 1 章　统计思维与网站分析 …… 1

1.1　不确定与确定 …… 1
1.2　统计分析方法 …… 4
1.2.1　细分分析 …… 4
1.2.2　对比分析 …… 6
1.2.3　趋势分析 …… 7
1.3　网站分析概要 …… 7
1.3.1　解决用户需求 …… 7
1.3.2　寻找新的用户需求 …… 10

第 2 章　R 语言数据操作基础 …… 11

2.1　R 简介 …… 11
2.2　了解 R 软件 …… 11
2.2.1　软件安装 …… 11
2.2.2　R 软件界面 …… 12
2.2.3　工作目录 …… 14
2.2.4　命令行交互 …… 15
2.2.5　命令脚本文件 …… 16
2.2.6　工作空间数据 …… 16
2.2.7　帮助 …… 17
2.2.8　R 语言入门 …… 18
2.2.9　扩展算法包 …… 19
2.3　R 语言的数据类型 …… 21
2.4　对象及其属性 …… 23
2.4.1　固有属性：模式和长度 …… 24
2.4.2　读取和设置属性值 …… 24
2.4.3　对象的搜索和删除 …… 25
2.5　向量 …… 26
2.5.1　创建向量 …… 26
2.5.2　向量索引 …… 26
2.5.3　向量编辑 …… 29
2.5.4　向量排序 …… 29
2.5.5　向量去重 …… 30
2.5.6　缺失值处理 …… 30
2.5.7　向量间操作 …… 31
2.6　矩阵和数组 …… 31
2.6.1　创建矩阵 …… 32
2.6.2　矩阵索引 …… 33
2.6.3　矩阵编辑 …… 34
2.6.4　矩阵的运算 …… 35
2.6.5　数组 …… 37
2.6.6　apply 函数应用 …… 38
2.7　列表和数据框 …… 39
2.7.1　列表的创建和索引 …… 39
2.7.2　列表编辑 …… 40
2.7.3　数据框的创建和名称 …… 40
2.7.4　数据框索引 …… 41
2.7.5　数据框编辑 …… 43

2.7.6 缺失值处理 …… 44
2.8 因子 …… 44
2.8.1 无序和有序因子 …… 45
2.8.2 连续数据的离散化 …… 46
2.9 字符串操作 …… 47
2.9.1 字符串长度 nchar …… 48
2.9.2 字符串合并和分割 …… 48
2.9.3 字符串内部字符的读取和替换 …… 49
2.9.4 正则表达式 …… 52
2.10 常用数据的创建 …… 53
2.10.1 因子序列的创建 …… 53
2.10.2 等差序列的创建 …… 54
2.10.3 随机抽样 sample …… 54
2.10.4 重复序列 rep …… 55
2.10.5 概率分布 …… 55
2.11 控制流 …… 58
2.11.1 分支语句 …… 58
2.11.2 循环语句 …… 59
2.12 运算符、函数和过程 …… 61
2.12.1 常用运算符 …… 61
2.12.2 函数 …… 62
2.12.3 过程 …… 63
2.13 数据的读写等操作 …… 64
2.13.1 读取数据 …… 64
2.13.2 输出数据 …… 68

第 3 章 R 语言的绘图基础 …… 69

3.1 概述 …… 69
3.2 颜色以及文字 / 点 / 线参数的设置 …… 69
3.2.1 颜色 …… 71
3.2.2 文字元素相关参数设置 …… 76
3.2.3 点元素相关参数设置 …… 77
3.2.4 线元素相关参数设置 …… 79
3.3 低级绘图函数 …… 80
3.3.1 引例 …… 80
3.3.2 标题 …… 81
3.3.3 图例 …… 83
3.3.4 坐标轴 …… 84
3.3.5 边框 …… 88
3.3.6 网格线 …… 89
3.3.7 点 …… 90
3.3.8 线 …… 91
3.3.9 文字 …… 94
3.3.10 多边形 …… 96
3.4 高级绘图函数以及常用绘图应用 …… 96
3.4.1 图形类型的选择 …… 96
3.4.2 散点图 …… 98
3.4.3 气泡图 …… 100
3.4.4 曲线图 …… 102
3.4.5 柱状图 …… 105
3.4.6 条形图 …… 109
3.4.7 饼图 …… 111
3.4.8 面积堆积图 …… 115
3.4.9 直方图和密度曲线图 …… 118
3.5 绘图窗口操作函数 …… 120
3.5.1 单一窗口中的子绘图区域布局 …… 120
3.5.2 绘图窗口操作 …… 120

第 4 章 单指标分析 …… 121

4.1 指标描述 …… 121
4.1.1 平均值和集中趋势 …… 121
4.1.2 正态分布 …… 126

4.1.3 频数分析 …… 130
4.1.4 描述性分析指标 …… 131
4.2 异常点监控 …… 133
4.2.1 概述 …… 133
4.2.2 P 控制图：监控转化率型指标 …… 135
4.2.3 单值 – 均值控制图 …… 142
4.2.4 单值 – 移动极差控制图 … 147
4.3 连续型指标的对比 …… 150
4.3.1 数据变换 …… 150
4.3.2 假设检验 …… 152
4.3.3 相同指标内的两组数对比：T 检验 …… 153
4.3.4 相同指标内的多组数据对比：单因素方差分析 …… 156
4.3.5 单因素协方差分析 …… 163
4.4 分类型指标的对比 …… 167
4.4.1 列联表分析 …… 167
4.4.2 卡方独立性检验 …… 172

第 5 章 时间序列分析 …… 177

5.1 时间序列 …… 177
5.2 增长率 …… 180
5.2.1 环比增长率 …… 180
5.2.2 同比增长率 …… 181
5.3 移动平均 …… 181
5.3.1 数学原理概述 …… 181
5.3.2 filter 函数 …… 182
5.3.3 R 语言实现 …… 184
5.4 指数平滑 …… 185
5.4.1 一次指数平滑 …… 186
5.4.2 二次指数平滑 …… 187
5.4.3 三次指数平滑 …… 191
5.5 ARIMA 模型 …… 194
5.5.1 自相关性 …… 194
5.5.2 平稳性和白噪声 …… 199
5.5.3 MA 滑动平均过程 …… 200
5.5.4 AR 自回归过程 …… 202
5.5.5 ARMA 自回归滑动平均混合过程 …… 203
5.5.6 检验模型质量 …… 205
5.5.7 非平稳时间序列的 ARIMA 过程 …… 208

第 6 章 连续指标建模：回归分析 … 213

6.1 一元线性回归分析 …… 213
6.1.1 引例 …… 213
6.1.2 一元线性回归分析的原理及 R 语言实现 …… 214
6.2 多元回归分析 …… 221
6.2.1 引例 …… 222
6.2.2 多元线性回归分析建模 … 222
6.2.3 模型修正函数 update() … 223
6.2.4 逐步回归分析函数 step() …… 226
6.2.5 自变量中包含分类型数据的回归分析 …… 228
6.3 Logic 回归分析 …… 230
6.3.1 引例及数据 …… 230
6.3.2 logic 分析的原理 …… 230
6.3.3 R 语言实现 …… 232
6.4 回归树 CART …… 237
6.4.1 rpart 函数 …… 237
6.4.2 预测及模型性能衡量 …… 240
6.4.3 过度拟合和剪枝 …… 240

第 7 章　分类指标建模：分类分析 ····· 243

7.1　决策树分类分析 ····· 243
7.1.1　概述 ····· 243
7.1.2　C4.5 算法 ····· 247
7.1.3　CART 算法 ····· 258
7.1.4　条件推理决策树算法 ····· 265
7.1.5　随机森林算法 ····· 267
7.2　贝叶斯分类 ····· 268
7.2.1　贝叶斯定理 ····· 268
7.2.2　朴素贝叶斯分类器 ····· 270
7.3　支持向量机 SVM ····· 271
7.3.1　原理 ····· 271
7.3.2　在 R 语言中实现非线性 SVM 分析 ····· 273
7.4　人工神经网络 ····· 274
7.4.1　神经元 ····· 274
7.4.2　两层网络 ····· 275
7.4.3　反向传播算法 ····· 276
7.4.4　R 语言实现 ····· 282
7.4.5　隐藏层中神经单元数目的确定 ····· 283
7.5　分类器的性能评估 ····· 284
7.5.1　混淆矩阵 ····· 284
7.5.2　ROC 曲线和 AUC ····· 287
7.5.3　提升度和提升曲线 ····· 291
7.5.4　洛伦兹曲线 ····· 292

第 8 章　样本细分 ····· 293

8.1　数据降维 ····· 293
8.1.1　问题引入 ····· 293
8.1.2　因子分析概述 ····· 294
8.1.3　factanal 函数 ····· 296
8.1.4　实例：问卷调查的因子分析 ····· 297
8.2　聚类分析 ····· 303
8.2.1　距离 ····· 303
8.2.2　层次聚类 ····· 313
8.2.3　kmeans 聚类 ····· 318
8.3　样本判别 ····· 321
8.3.1　knn（k 最近邻分类）算法 ····· 323
8.3.2　实例：基于 knn 算法的商品推荐系统 ····· 324

第 9 章　样本间的关系 ····· 329

9.1　关联分析 ····· 329
9.1.1　关联分析的原理 ····· 329
9.1.2　Apriori 算法的 R 语言实现 ····· 331
9.2　序列模式关联分析 ····· 334
9.2.1　序列模式关联分析的原理 ····· 334
9.2.2　序列模式关联分析的 R 语言实现 ····· 336
9.2.3　实例：用户点击页面的行为分析 ····· 338
9.3　关系网络分析 ····· 346
9.3.1　网络关系分析的基本概念 ····· 346
9.3.2　中心度 ····· 356
9.3.3　中心势 ····· 362
9.3.4　社群发现 ····· 363
9.3.5　实例：分析微博的传播特性 ····· 367
9.3.6　实例：购物车商品分类分析 ····· 370

第 10 章　文本分析 ····· 378

10.1　数据处理 ····· 378

10.1.1　数据引例 ····· 378

10.1.2　分词和词条 ····· 380

10.1.3　语料库和文档 ····· 381

10.1.4　词条—文档关系矩阵 ····· 384

10.2　实例：QQ 群消息历史分析 ····· 387

10.2.1　数据集 ····· 387

10.2.2　数据集格式转化 ····· 389

10.2.3　留言文本的分词操作 ····· 390

10.2.4　分析常用话题词汇 ····· 391

10.2.5　建立用户－词条间的网络关系图 ····· 392

10.2.6　绘制重点词条和用户的网络图 ····· 392

10.3　XML 包爬取网页数据 ····· 394

10.3.1　htmlParse 函数 ····· 395

10.3.2　getNodeSet 函数 ····· 395

10.3.3　xmlValue 函数 ····· 396

10.3.4　xmlGetAttr 函数 ····· 396

10.3.5　实例：爬取某电商网站衬衫类目商品的部分信息 ····· 397

第 11 章　网站指标监控系统的搭建 ····· 400

11.1　gWidgets 包基础 ····· 400

11.1.1　环境搭建 ····· 400

11.1.2　引例 ····· 401

11.1.3　常用控件 ····· 406

11.2　实例：商品搜索算法参数配置监控 GUI 系统 ····· 414

11.2.1　商品搜索算法概述 ····· 414

11.2.2　原始商品信息 ····· 415

11.2.3　系统概述 ····· 417

11.2.4　模块布局 ····· 418

11.2.5　计算除关键字匹配以外的指标得分 ····· 420

11.2.6　模拟一次搜索行为数据 ··· 421

11.2.7　模拟多次搜索行为数据 ··· 423

11.2.8　绘图 ····· 423

第 12 章　基于 RFM 模型的客户价值系统 ····· 427

12.1　马尔科夫链 ····· 427

12.1.1　引例 ····· 427

12.1.2　概念 ····· 428

12.1.3　预测实例 ····· 428

12.2　RFM 模型 ····· 431

12.2.1　RFM 的概念 ····· 431

12.2.2　数据集 ····· 431

12.2.3　购买行为随机模型 ····· 432

12.3　结合马尔科夫链的 RFM 模型 ····· 436

12.3.1　购买状态转移矩阵 ····· 437

12.3.2　预测用户购买状态 ····· 445

第 1 章

统计思维与网站分析

第 1 章先大略地谈谈数据统计和分析从业人员的常用思维方式（理论），以及网站数据分析和挖掘的整体思路，希望能帮助读者从数学思维和统计理论上对全书有整体的认识。

1.1 不确定与确定

日常生活中的很多自然规律都是具有确定性结果的。也就是说，在多次相同的实验环境下，其实验结果均是相同的。例如，在相同的位置观察 100 天内太阳升起的方向，如果从东方升起则记为 T，否则记为 F。那么被记录下来的 100 条数据一定均为 T，实际上太阳每天都是从东方升起的。这种类型的案例就是有确定性结果的案例。那什么是具有不确定性结果的案例呢？例如，以相同的方式投掷同一枚硬币，共实验 100 次，如果正面向上则记为 T，反之记为 F。在这种情况下，最终的数据是不能预测的，实际上每次投掷的结果既可能是正面，又可能是反面，这就是一个不确定性结果的案例。

对于有确定性变量的，可以使用数学公式来抽象出其中的规律，从而在已知某些变量数据的情况下，预测另一些变量的数值或范围。在上面的例子中，“太阳是否从东方升起”这个变量的取值恒为 T，可以使用一个数值来概括该确定性变量。而对于“投掷硬币结果为正面”这个变量的取值就无法简单地使用一个数值来确定了，因为每次实验的结果均有可能不同，那又该如何研究这类不确定的变量呢？

继续来做投掷硬币的实验，首先投掷 10 次并记录下结果为正反两面的次数。然后投掷 20 次、30 次，直至 2500 次并依次记录结果，其统计结果如表 1-1 所示。

表 1-1　投掷实验统计结果

实验 id	投掷次数	正面次数	反面次数	正面次数 / 投掷次数
实验 1	10	7	3	7/10=0.7
实验 2	20	12	8	12/20=0.6
…	…	…	…	…

（续）

实验 id	投掷次数	正面次数	反面次数	正面次数 / 投掷次数
实验 249	2490	1314	1176	1314/2490=0.5277108
实验 250	2500	1259	1241	1259/2500=0.5036

在该统计表的基础上，以投掷次数为横轴，正面次数 / 投掷次数的比例为纵轴，绘制出曲线图，如图 1-1 所示。

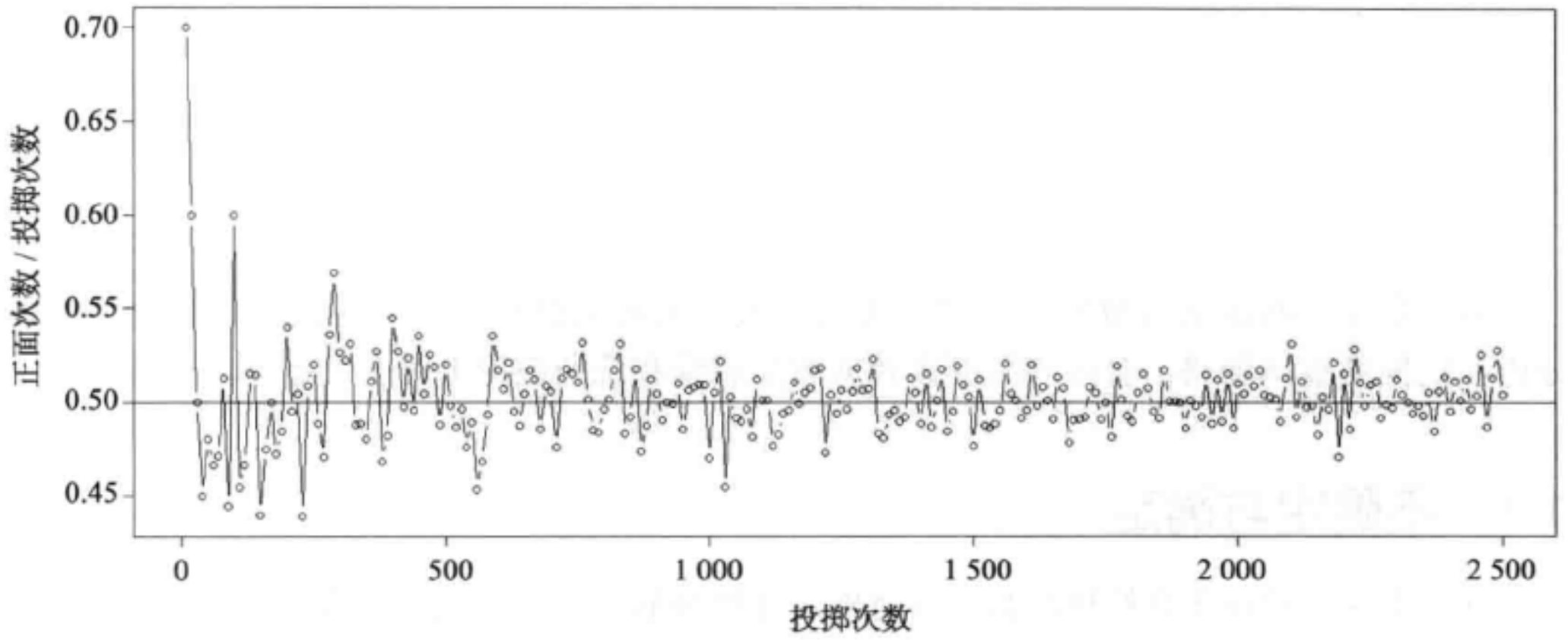


图 1-1　投掷次数 – 正反面比例散点图

观察图形得知：随着投掷次数逐步增加，正面次数 / 投掷次数在 0.5 上下浮动，并且浮动的幅度越来越小，即其不确定性越来越小。可见，虽然我们无法抽象出某一次投掷硬币的结果，但是有可能在大量投掷次数的情况下，抽象出正面次数 / 投掷次数的结果。下面研究投掷次数处于 [2000，2500] 范围内，变量“正面次数 / 投掷次数”的结果，如表 1-2 所示。

表 1-2　投掷实验结果数据

实验 id	投掷次数	正面次数	反面次数	正面次数 / 投掷次数
实验 1	2000	989	1011	989/2000=0.4945
实验 2	2001	1003	998	1003/2001=0.5012494
…	…	…	…	…
实验 500	2499	1233	1266	1233/2499=0.4933974
实验 501	2500	1250	1250	1250/2500=0.5

绘制出的曲线图如图 1-2 所示。

观察图 1-2 可知：变量“正面次数 / 投掷次数”的不确定性已经很低了，它明显趋近于数值 0.5。下面统计该变量在 [0.465，0.535] 范围内并以 0.005 为间隔的数值，如表 1-3 所示。

把以上数据绘制成直方图（称为频数统计图），如图 1-3 所示。

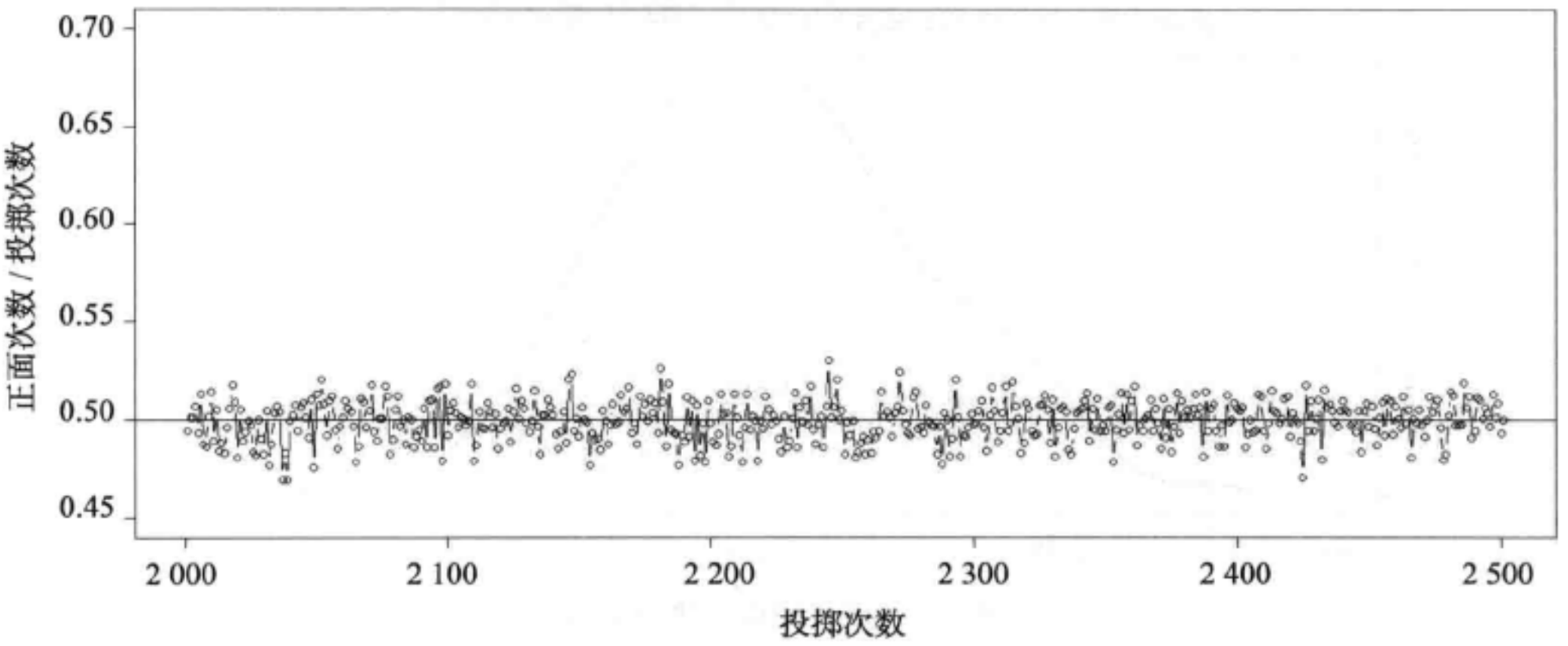


图 1-2 [2000，2500] 范围内投掷次数 – 正反面比例散点图

表 1-3 实验结果的频数统计数据

范围	次数统计	范围	次数统计
[0.465，0.47）	2	[0.5，0.505）	90
[0.47，0.475）	1	[0.505，0.51）	80
[0.475，0.48）	15	[0.51，0.515）	52
[0.48，0.485）	31	[0.515，0.52）	19
[0.485，0.49）	45	[0.52，0.525）	6
[0.49，0.495）	70	[0.525，0.53）	1
[0.495，0.5）	88	[0.53，0.535）	1

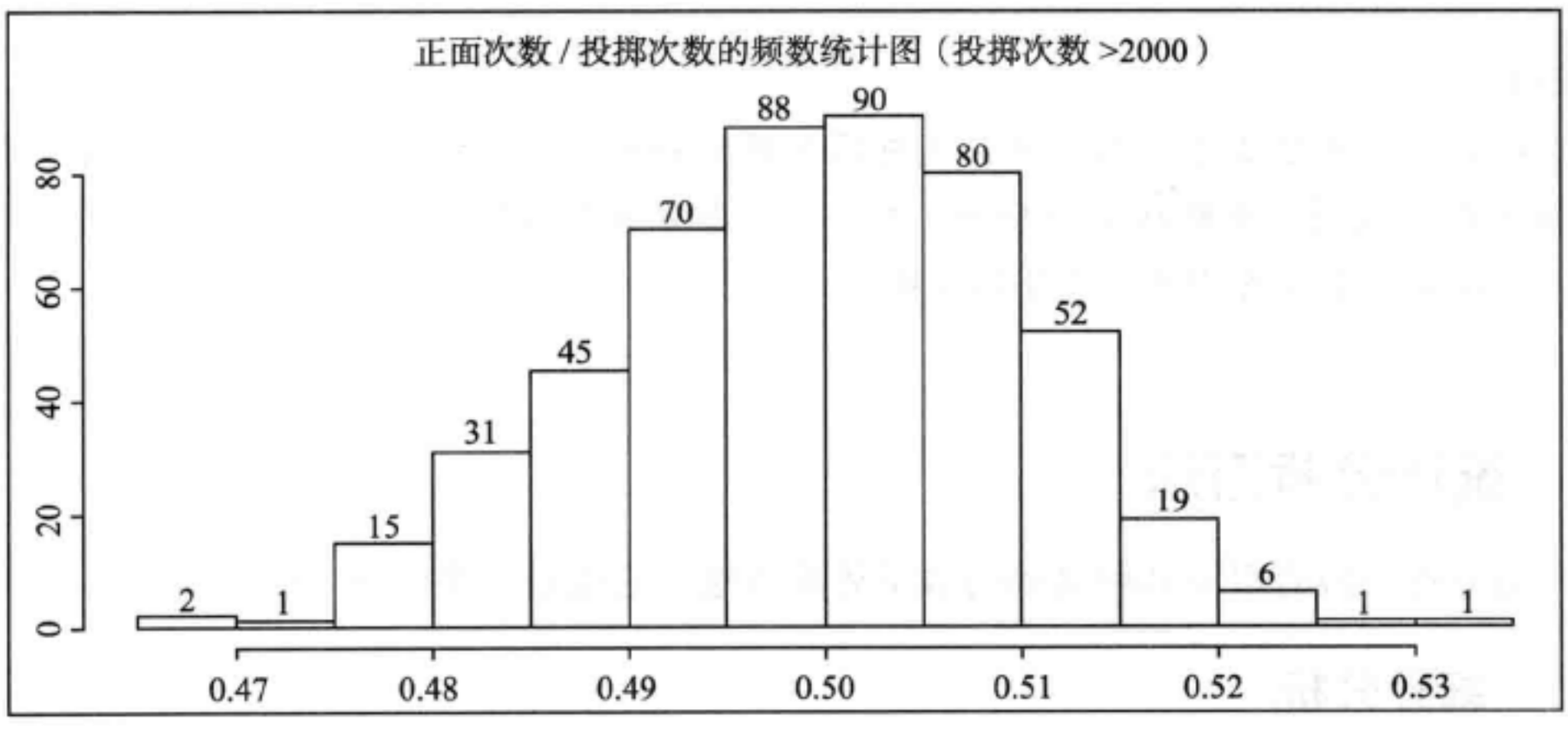


图 1-3 实验结果的频数统计直方图

如果经过足够多的实验后得到大量数据，频数统计图中横轴的间隔足够小，则可以得到一条平滑曲线（称为概率密度曲线），如图 1-4 所示。

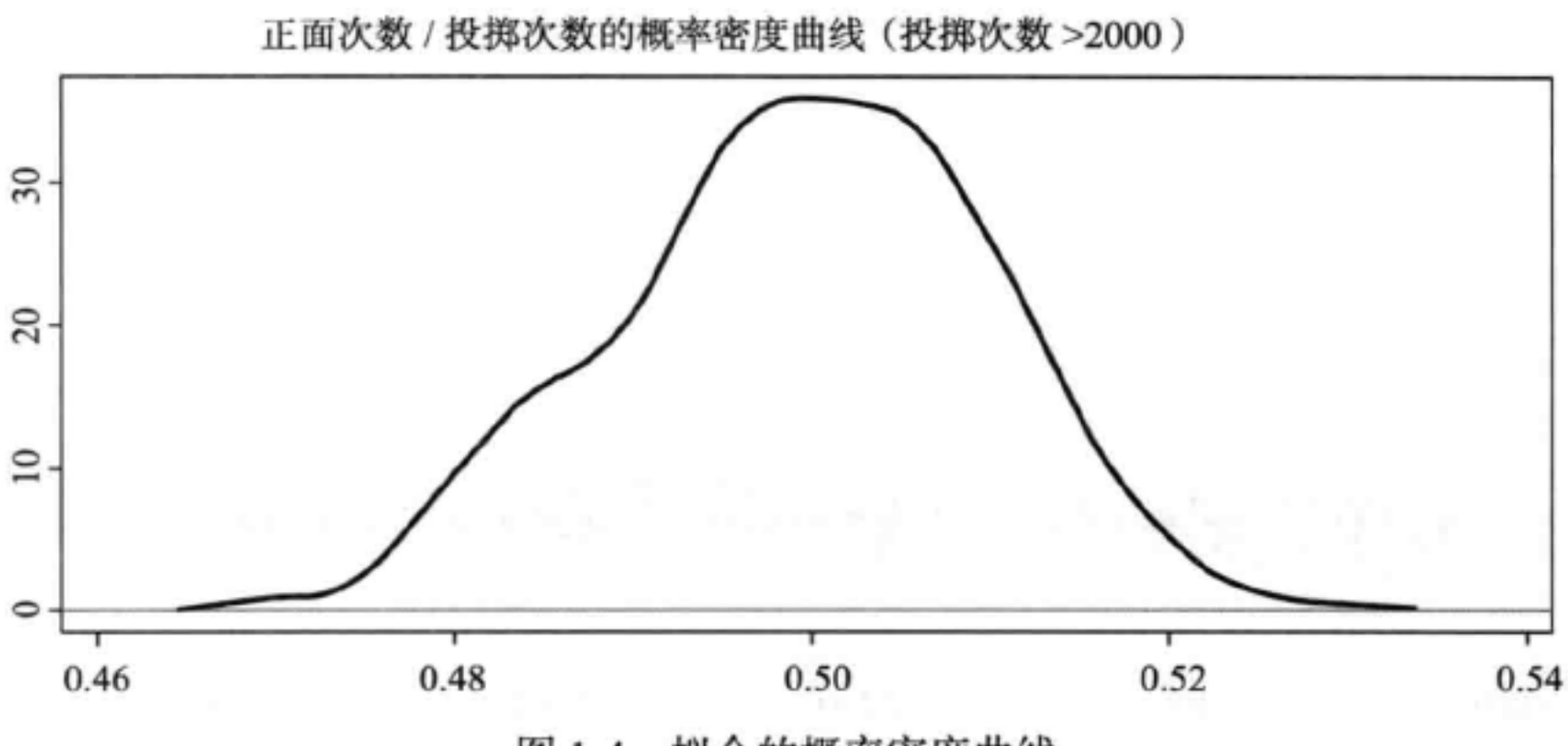


图 1-4 拟合的概率密度曲线

该曲线类似于钟形曲线，统计学上多用正态分布曲线的分布规律拟合，也就是说，该类变量服从正态分布。现代统计学中的很多理论均是在正态分布的基础上发展起来的。一旦某一变量服从正态分布，就可以使用对应的统计学理论来预测该不确定性变量的预测估值及其范围，并可以给出该预测值的可信度。

在网站分析中相关的应用也很多。例如，在研究用户每天浏览网站的情况时，由于不同用户是否会浏览的可能性均不同，因此无法预测“某一个体每天的浏览行为”，这类似于上文提及的“某一次投掷硬币是否为正面”的问题。但是可以统计每天大量用户浏览的次数，因为该值不确定性的可能性很低，这类似于上文提及的“大量投掷次数的情况下，研究正面次数 / 投掷次数”的问题。如果该变量服从了正态分布，就可以使用相应的理论从整体上研究用户的浏览行为。

注意

在统计学中除了正态分布以外，还有很多典型分布，这里暂不一一介绍。网站统计分析的基础就是：找到一个服从某种分布（主要是正态分布）的变量，用以衡量网站对绝大部分用户（而非某一个特定用户）的整体性能。

1.2 统计分析方法

网站分析一般有以下几种基础的数学思维方法，这也是大多数统计分析的方法。

1.2.1 细分分析

由上文可知，网站分析的基础是找出服从正态分布（或其他分布）的变量，其中，细分分析是最主要的方式。何为细分？来看个案例，图 1-5 是某网站两年内日浏览量（PV）的频数统计图。

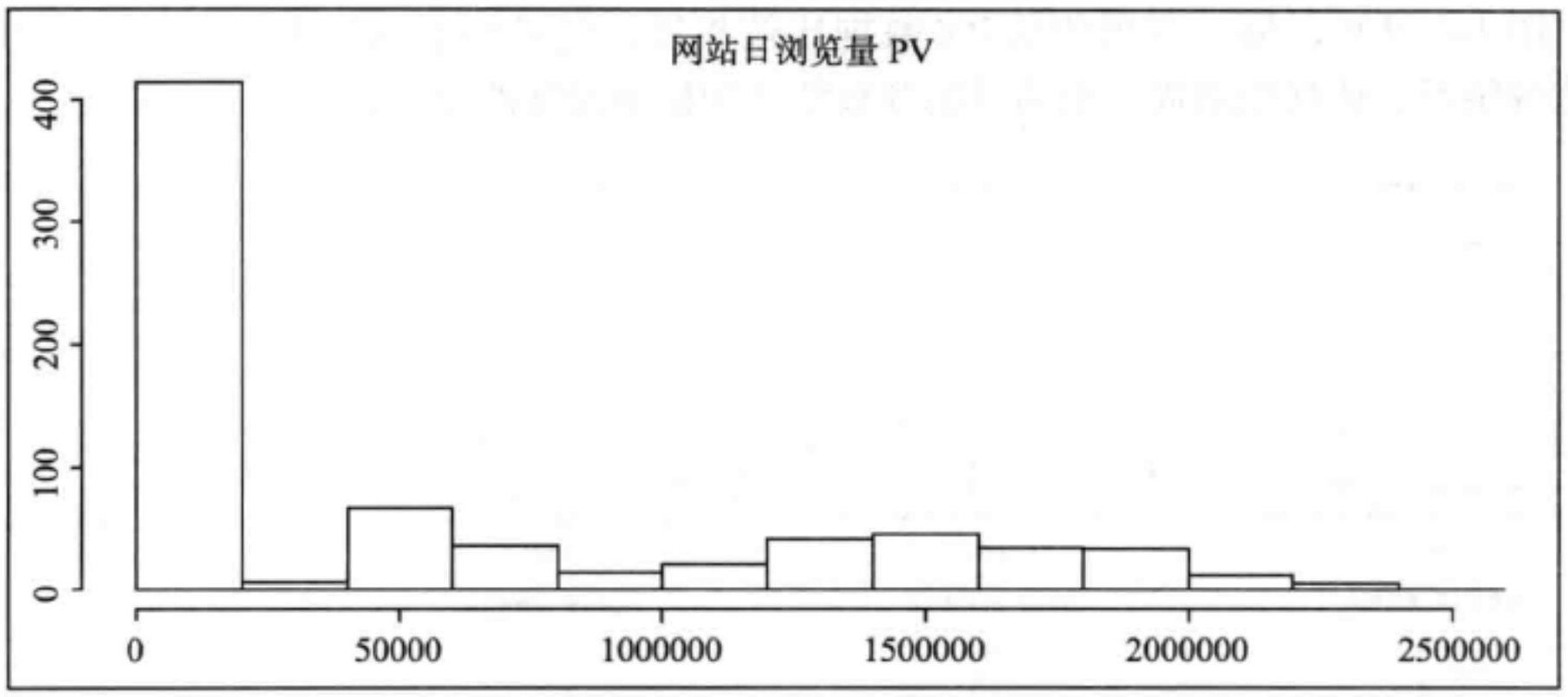


图 1-5 网站两年内日浏览量的频数统计图

由图 1-5 可知，该网站的 PV 主要集中在 0 ～ 300000 之间，但它并不服从任何典型的分布。此时就要思考是否会有其他因素影响了该变量，如用户的自然属性（性别、学历、职业等）或是否为周末等因素。既然如此，那接下来试着以天为单位，看看周一至周日每天的 PV 情况，如表 1-4 所示。

表 1-4 网站两年内的日浏览量数据

	周一	周二	周三	周四	周五	周六	周日
1	105090	171311	157888	111815	654991	1475446	1827615
2	122937	177996	143907	131283	479044	544074	2051236
…	…	…	…	…	…	…	…
104	122907	175668	141378	129122	665340	1170877	1849763

然后把每行数据（表示每周的数据）绘制为一条折线（见图 1-6），这里共绘制出了 104 条折线。

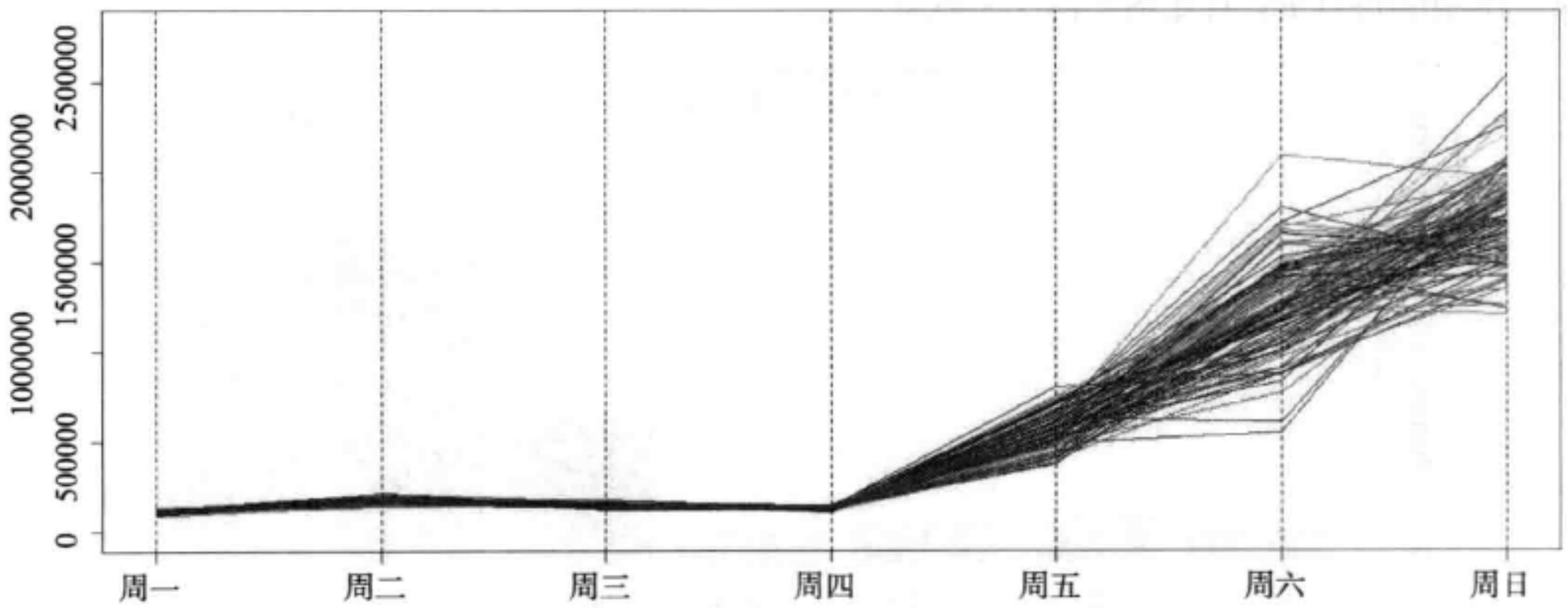


图 1-6 每周 PV 折线图

由图 1-6 可知，周一至周四的 PV 数据比较相近，但是周五和周末的 PV 则明显不同，基于这种情况，依次绘制周一至周日的频数统计和概率密度曲线图，如图 1-7 所示。

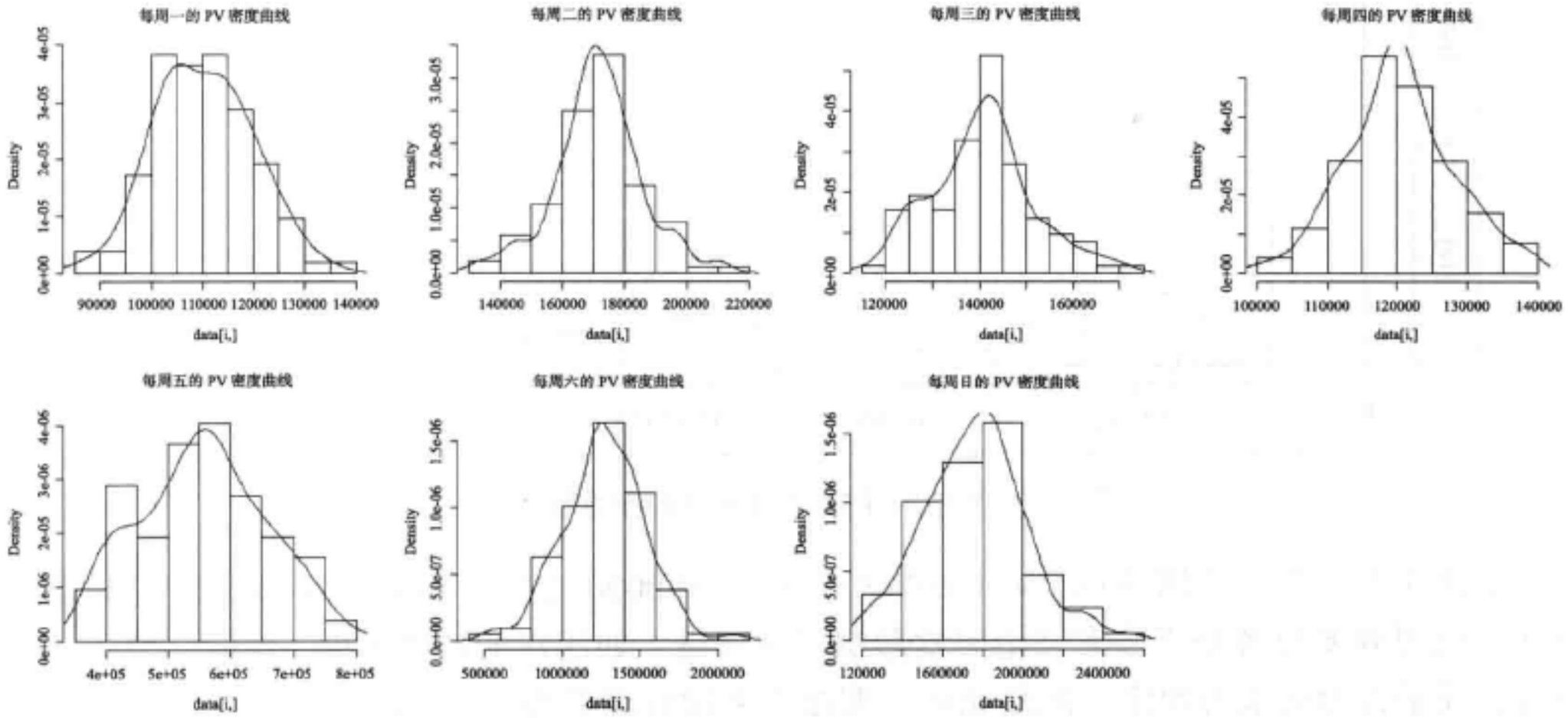


图 1-7 周一至周日的日浏览量

图 1-7 中已经呈现出了钟形图，实际上经过计算它们均服从正态分布。上述问题的关键就是使用了某一特定因素把原始变量细分为了多个变量（分为了 7 个，依次表示周一至周日的不同 PV），即采用细分方式来找出规律。由于细分变量服从正态分布，所以可以使用相应理论进一步研究。

1.2.2 对比分析

在细分分析后，可以进一步使用对比分析来找到指标变量变化的真实原因。在上例中，由于周一至周日的 PV 各自服从正态分布，所以可以使用其均值来代表网站的整体 PV 性能，绘制的每日 PV 对比图如图 1-8 所示。

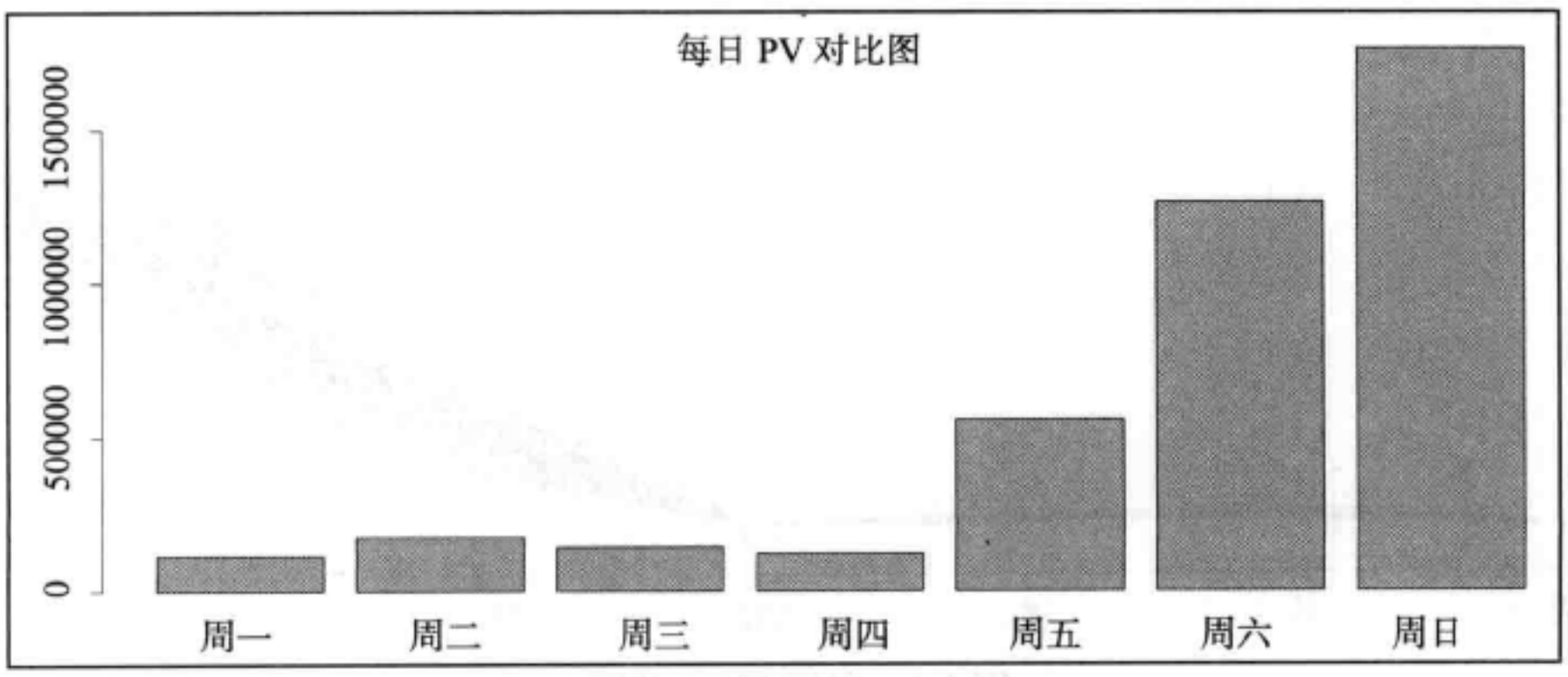


图 1-8 每日 PV 对比图

从图 1-8 中可以分析出大部分用户喜欢在周末浏览网站，并且周日浏览的可能性最高。经过对比分析，可以找到究竟是哪些因素影响了指标，其影响的幅度各有多大，从而帮助决策者抓住问题的本质。

1.2.3 趋势分析

趋势分析主要是研究指标变量的上升下降，以及周期性等趋势性分析指标的异常变动。如果网站是新建立的并处于上升期，则其日 PV 就不会一直是平稳的。图 1-9 是某网站成立两年内每周六的 PV 趋势变化曲线。

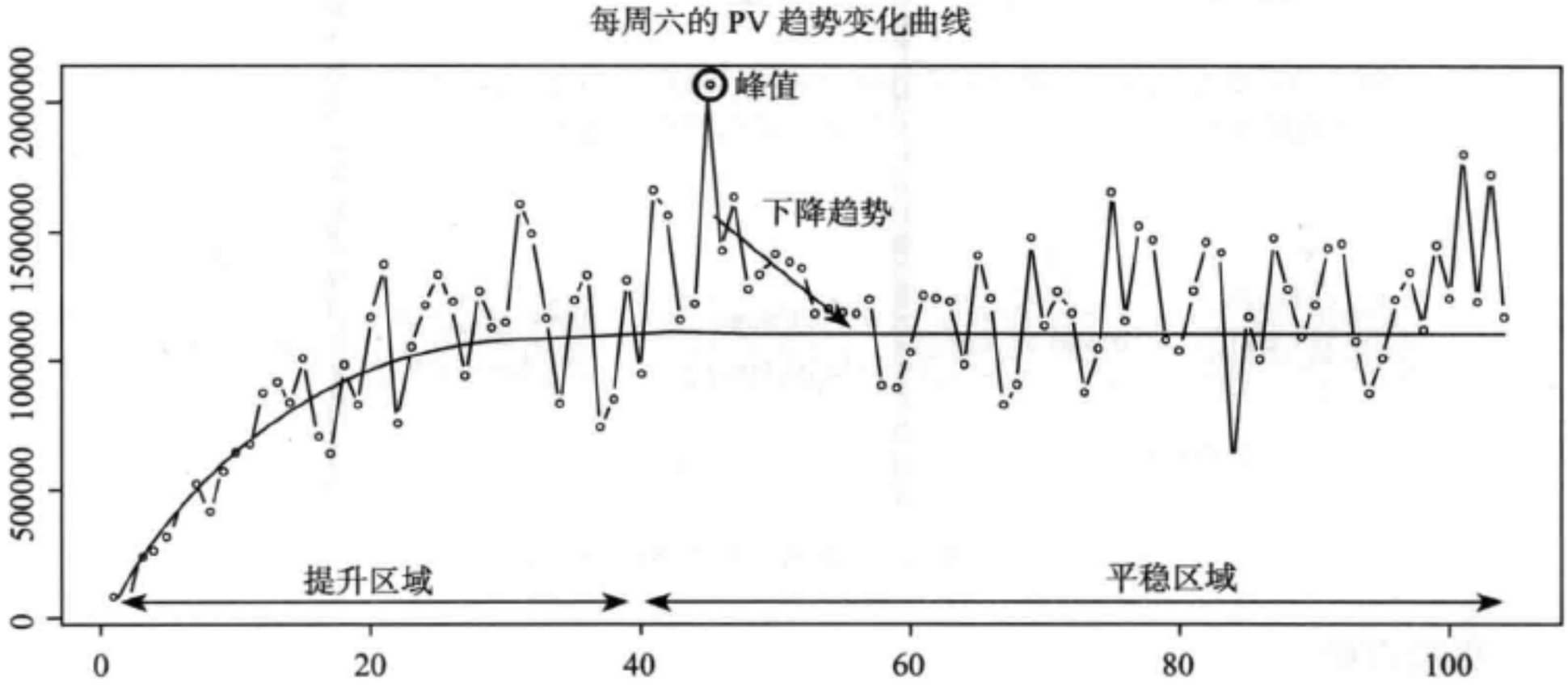


图 1-9 每周六的 PV 趋势变化曲线

可见网站在成立约 40 周后趋于平稳，在第 47 周出现了峰值并在之后的 10 周内出现了明显的下降趋势。使用趋势分析，可以观察出指标的长期走势，从而帮助决策者预测日后的指标变化，通过实时监控指标的短期变化，也可以尽早察觉指标的异常变动情况，从而尽早发现问题。

1.3 网站分析概要

1.3.1 解决用户需求

网站存在的意义是解决部分访客（目标群体）的某些需求。因此网站分析的主要目的是通过量化指标来衡量网站解决用户需求的能力，其主线可以分为如下 3 个阶段：

1）有多少访客来到了网站（投入），可使用引流类指标来衡量。

2）网站对访客的黏性，它体现了用户在通过网站寻求解决自身需求的过程中，所表现出来的整体性能情况，可使用黏性指标来衡量。

3）网站最后解决了多少用户的需求（产出或投入产出比），可使用产出指标来衡量。

图 1-10 通过和传统超市的对比，形象地体现了一个电子商务网站解决用户购买商品这一主要需求的过程，及其使用的主要指标。

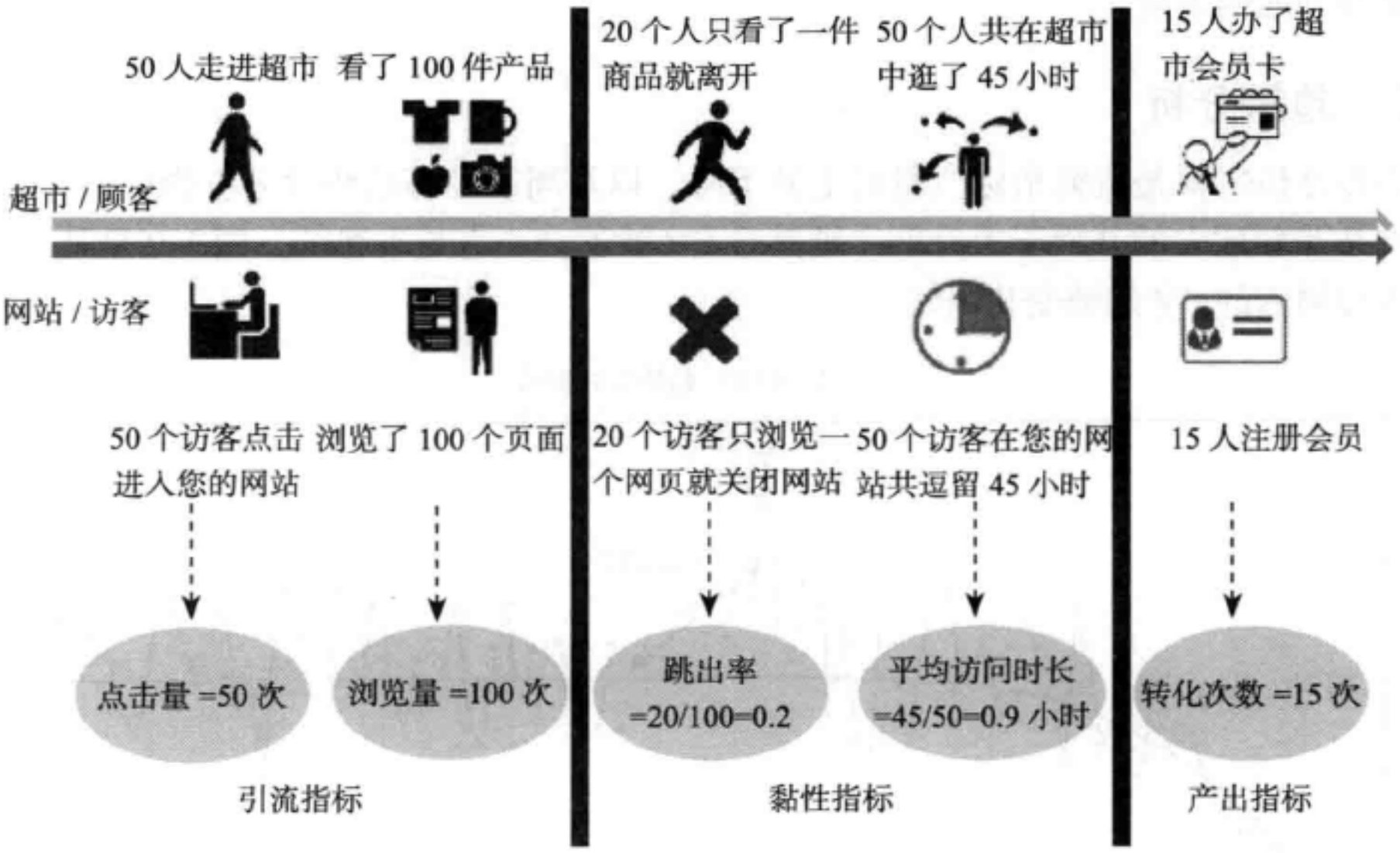


图 1-10[⊖] 电子商务网站用户购买商品的过程

1. 引流指标

引流指标包括浏览量（PV）、访问次数、访客数（UV）、新访客数、新访客比率、IP 等内容[⊜]。

（1）浏览量（PV）

定义：页面浏览量即为 PV（page view），用户每打开一个页面就被记录 1 次。

含义：PV 越多，说明该页面被浏览得越多。PV 之于网站，就像收视率之于电视，已成为评估网站表现的基本尺度。

（2）访问次数

定义：访问次数即 visit，是指访客在网站上的会话（session）次数，一次会话过程中可能浏览多个页面。

含义：页面浏览量（PV）是从页面角度衡量加载次数的统计指标，而访问次数（visit）则是从访客角度衡量访问的分析指标。如果网站的用户黏性足够好，同一用户一天中多次登录网站，那么访问次数就会明显大于访客数。

（3）访客数（UV）

定义：访客数（UV）即唯一访客数，是指一天之内网站的独立访客数（以 Cookie 为依

⊖ 本图源于《百度统计网站分析白皮书》。

⊜ 此处的指标定义参考了《百度统计：网站分析白皮书》。

据），一天内同一访客多次访问网站只计算一个访客。

含义：唯一访客数是从访客角度看访客到达网站的数量。

（4）新访客数

定义：一天的独立访客中，历史第一次访问网站的访客数。

含义：新访客数可以衡量营销活动开发新用户的效果。

（5）新访客比率

定义：新访客比率 = 新访客数 / 访客数，即一天中新访客数占总访客数的比例。

含义：整体访客数不断增加，并且其中的新访客比例较高，能表现网站运营在不断进步。就像人体的血液循环一样，有新鲜的血液不断补充进来，充满活力。

（6）IP 数

定义：一天之内，访问网站的不同独立 IP 数之和。其中，同一 IP 无论访问了几个页面，独立 IP 数均为 1。

含义：从 IP 数的角度衡量网站的流量。

2. 黏性指标

黏性指标涉及跳出率、平均访问时长、平均访问页数等内容。

（1）跳出率

定义：只浏览了一个页面便离开网站的访问次数占总访问次数的百分比，即只浏览了一个页面的访问次数 / 全部的访问次数汇总。

含义：跳出率是非常重要的访客黏性指标，它显示了访客对网站的兴趣程度。跳出率越低，说明流量质量越好，访客对网站的内容越感兴趣，这些访客越可能是网站的有效用户、忠实用户。该指标也可以衡量网络营销的效果，指出有多少访客被网络营销吸引到宣传产品页或网站之后，又流失掉了。例如，网站在某媒体上打广告推广，分析从这个推广来源进入的访客指标，其跳出率可以反映出选择这个媒体是否合适，广告语的撰写是否优秀，以及网站入口页设计的用户体验是否良好。

（2）平均访问时长

定义：平均每次访问在网站的停留时长，即平均访问时长等于总访问时长与访问次数的比值。

含义：平均访问时间越长，说明访客停留在网页的时间越长。如果用户对网站的内容不感兴趣，则会较快关闭网页，那么平均访问时长就短；如果用户对网站的内容很感兴趣，在网站停留了很长时间，则平均访问时长就长。

（3）平均访问页数

定义：平均每次访问浏览的页面数量，平均访问页数 = 浏览量 / 访问次数。

含义：平均访问页数越多，说明访客对网站的兴趣越大。而浏览信息多也使得访客对网站更加了解，这对网站市场信息的传递、品牌印象的生成，以至于将来的销售促进都是有好处的。一般，会将平均访问页数和平均访问时长这两个指标放在一起分析，进而衡量网站的用户体验情况。

3. 产出指标

产出指标又包含了转化次数、转化率两个指标。

（1）转化次数

定义：访客到达转化目标页面，或完成网站运营者期望其完成动作的次数。

含义：转化就是访客做了任意一项网站管理者希望访客做的事，与网站运营者期望达到的推广目的和效果有关。

（2）转化率

定义：转化率 = 转化次数 / 访问次数。

含义：转化率即访问转化的效率，数值越高，说明越多的访问次数完成了网站运营者希望访客进行的操作。

1.3.2 寻找新的用户需求

除了要衡量网站解决用户需求的能力以外，网站分析人员还要通过数据指标来寻找新的用户需求，并对用户进行细分，最后实现网站对用户的个性化服务。以电子商务网站为例，一般可以通过如表 1-5 所示的途径来收集用户的新需求。

表 1-5 收集用户需求的途径

序号	说明
1	网站中某关键性网页或区域的点击量和使用量
2	用户对网站的建议留言或问卷调查结果
3	用户在站内搜索中输入的关键字
4	用户在站外搜索引擎中输入的关键字
5	用户对客服的留言以及商品的评价

此外，随着社交网络的兴起，用户的站外分享内容也是寻找用户需求的新途径。

在收集这些个体用户提出的需求后，要使用科学的统计方法来对其进行分析，并把不同的需求进行等级分类，同时要拒绝某些极少数个体提出的非合理需求。

关于网站的分析方法，本书只在这一章简要进行了讲解，对于具体方法和应用实践均会在后续章节中结合网站分析和 R 语言实现案例来体现。读者若想系统性地了解网站分析理论可以参考笔者的博客文章：http://www.bassary.com/?p=1451。

第 2 章

R 语言数据操作基础

2.1 R 简介

这里谈到的 R 涉及两个含义，它既表示一种用于数据分析建模以及绘图的语言，又指一个有着统计分析功能及强大作图功能的软件系统。R 语言是由新西兰奥克兰大学的 Ross Ihaka 和 Robert Gentleman 共同创建的，因此他们把该语言称为 R 语言。该语言在 GNU 协议 General Public Licence 4 下免费发行，现在由“R 开发核心团队”负责开发。由于 R 语言是基于由 AT&T 贝尔实验室所创的 S 语言发展出的一种语言，因此，通常情况下使用 S 语言编写的代码都可以不做修改地在 R 环境下运行。

R 软件是一个免费的自由软件，包括 UNIX、Linux、MacOS 和 Windows 等几个版本，可以免费下载 R 的安装程序、各种外挂程序和文档等。在 R 的安装程序中只包含 8 个基础模块，其他拓展的算法包可以通过 CRAN 获得。在 CRAN 中，除了有收藏 R 的执行档下载版、源代码和说明文件以外，还收录了各种用户撰写的软件包。目前，全球有超过 100 个 CRAN 镜像站。

2.2 了解 R 软件

2.2.1 软件安装

新用户可以通过 http://cran.r-project.org/mirrors.html 浏览 CRAN 镜像站地址，其中包含了中国大陆地区的 5 个镜像地址，如图 2-1 所示。

China	
http://ftp.ctex.org/mirrors/CRAN/	CTEX.ORG
http://mirror.bjtu.edu.cn/cran	Beijing Jiaotong University, Beijing
http://cran.dataguru.cn	Dataguru (a, Guangzhou
http://mirrors.ustc.edu.cn/CRAN/	University of Science and Technology of China
http://mirrors.xmu.edu.cn/CRAN/	Xiamen University

图 2-1　R 软件包镜像地址

点击上面的任何一个地址均可进入其镜像的详情页面，通过该页面，可以下载 Linux/Macos/Windows 操作系统的 R 软件安装软件包，如图 2-2 所示。

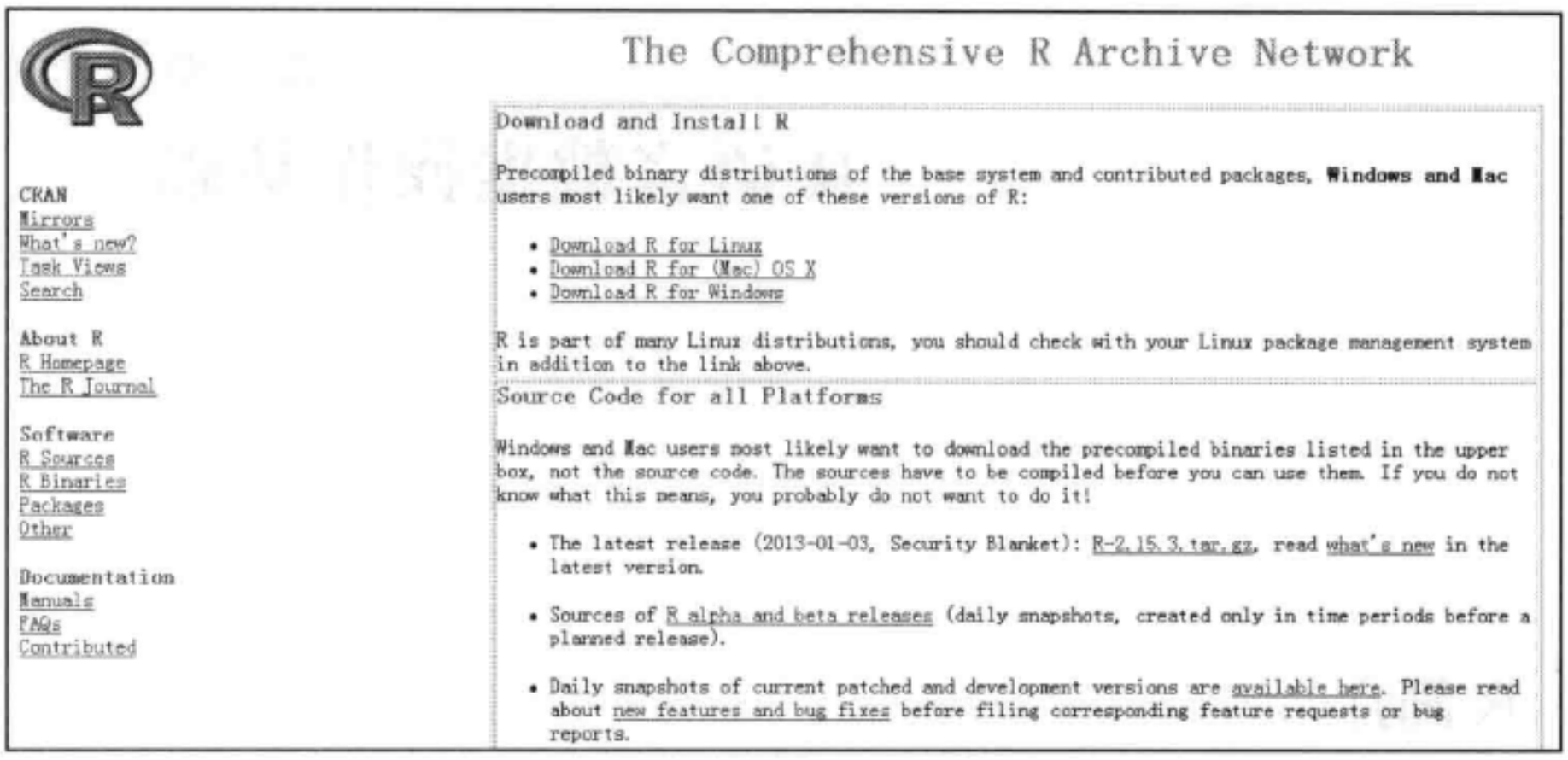


图 2-2 R 软件下载界面

2.2.2 R 软件界面

下载并安装 R 软件后，双击打开该软件，可以看到 R 软件界面，如图 2-3 所示。可以在命令行中直接输入 R 语言命令，运行并展示其结果。命令行最主要的两个应用是“文件”菜单（见图 2-4）和“程序包”菜单（见图 2-5）。

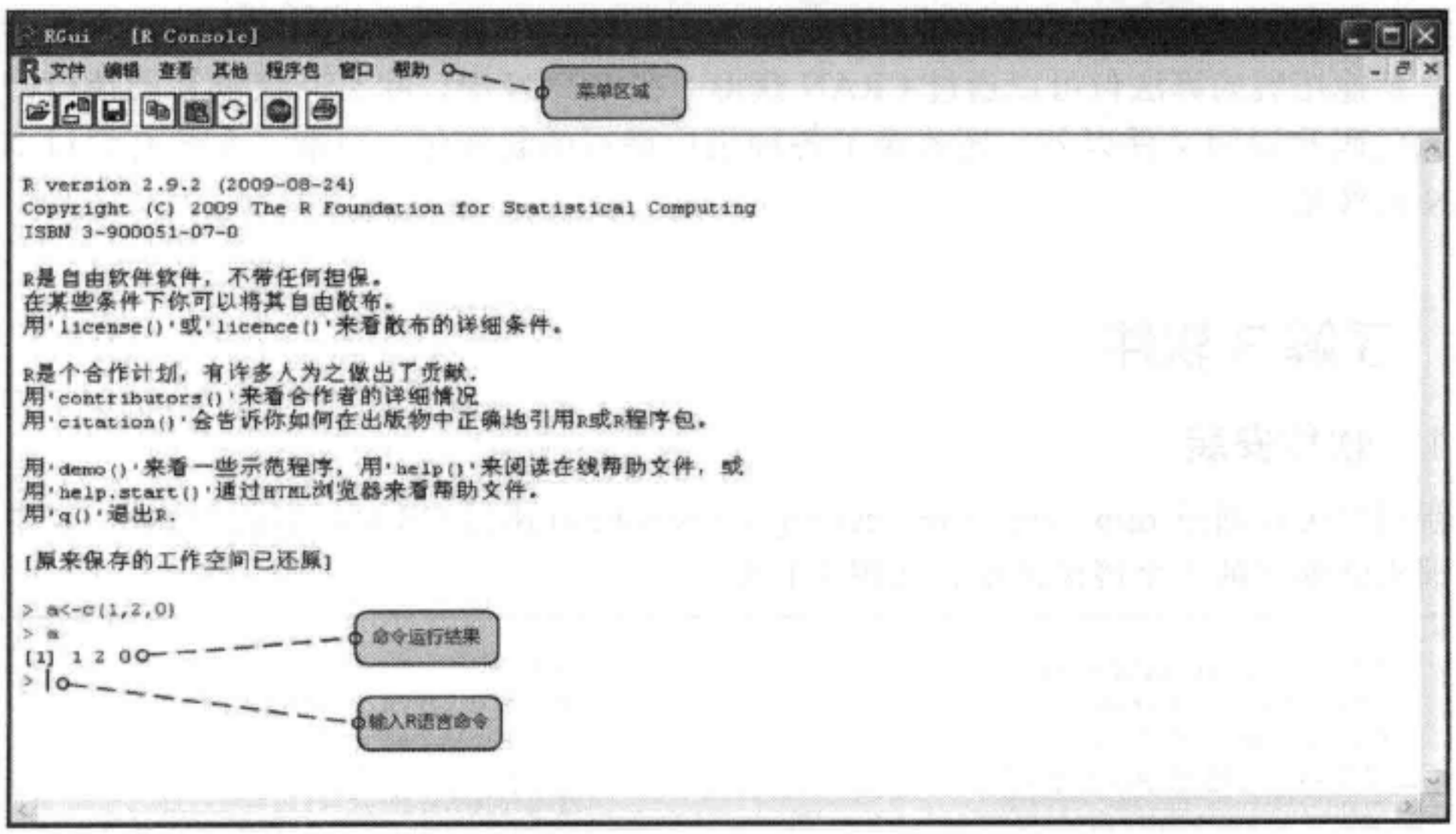


图 2-3 R 软件界面

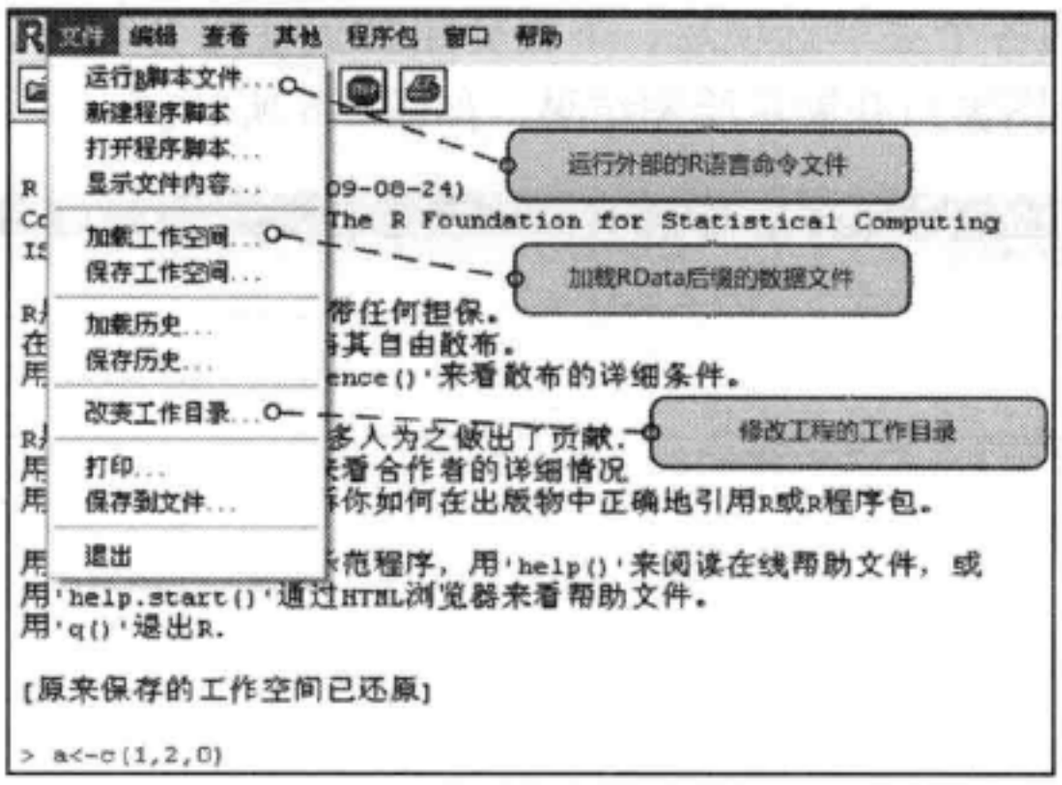


图 2-4 R 软件的“文件”菜单

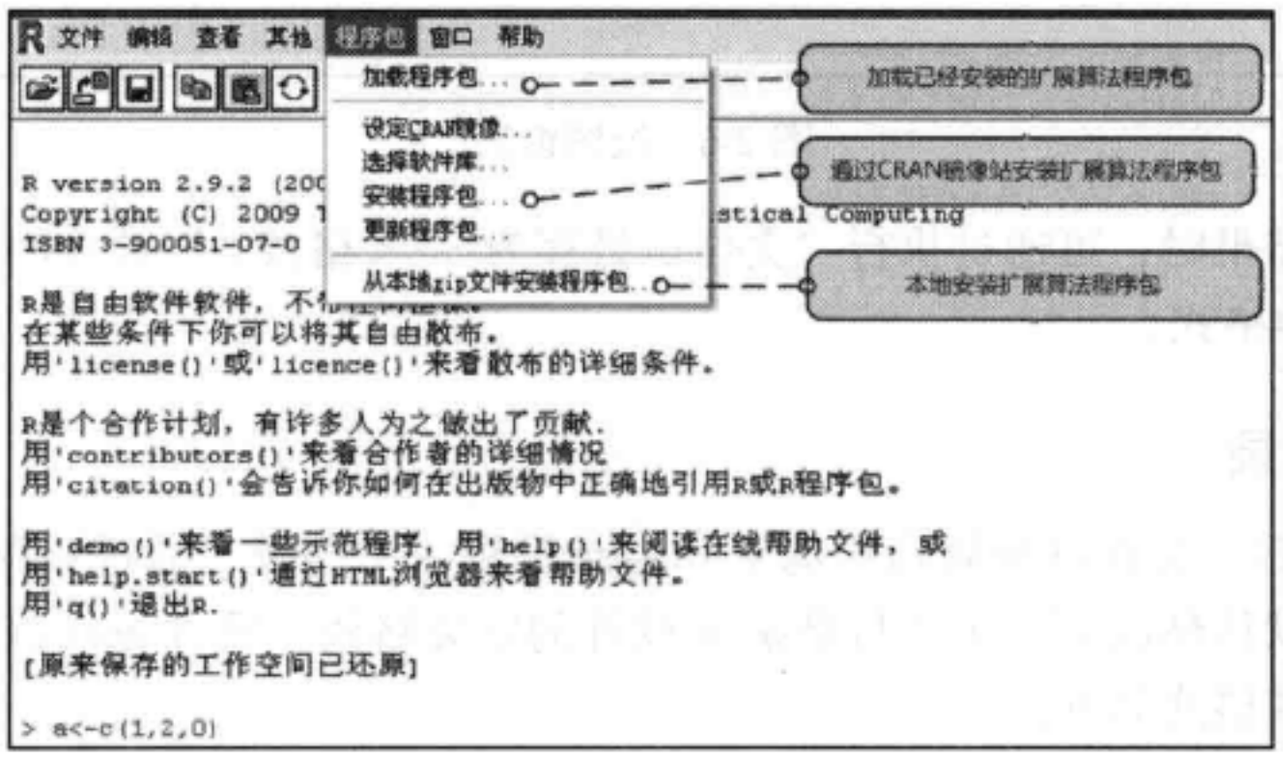


图 2-5 R 软件的“程序包”菜单

重点了解“文件”菜单中的如下选项：

- 运行 R 脚本文件：浏览本地硬盘中的 R 语言命令文件（多是以 .r 为后缀名的文件）。
- 加载工作空间：加载每次关闭程序前保留的数据文件（以 .RData 为后缀名的文件），也可以使用 load 命令来实现。
- 改变工作目录：修改工程目录。

“程序包”菜单中的主要选项如下：

- 加载程序包：选择已经安装到 R 软件中的扩展算法的程序包，也可以用 library 命令来实现。
- 安装程序包：通过 CRAN 镜像站下载并直接安装扩展算法的程序包，也可以用 install.packages 命令来实现。
- 从本地 zip 文件安装程序包：通过浏览本地硬盘中扩展算法的程序包源文件来安装算法包。

在了解了R软件的各个菜单选项后，下面介绍如何使用R软件，例如，要执行一条绘图命令，这时会弹出绘图窗口并展示绘图结果，如图2-6所示。

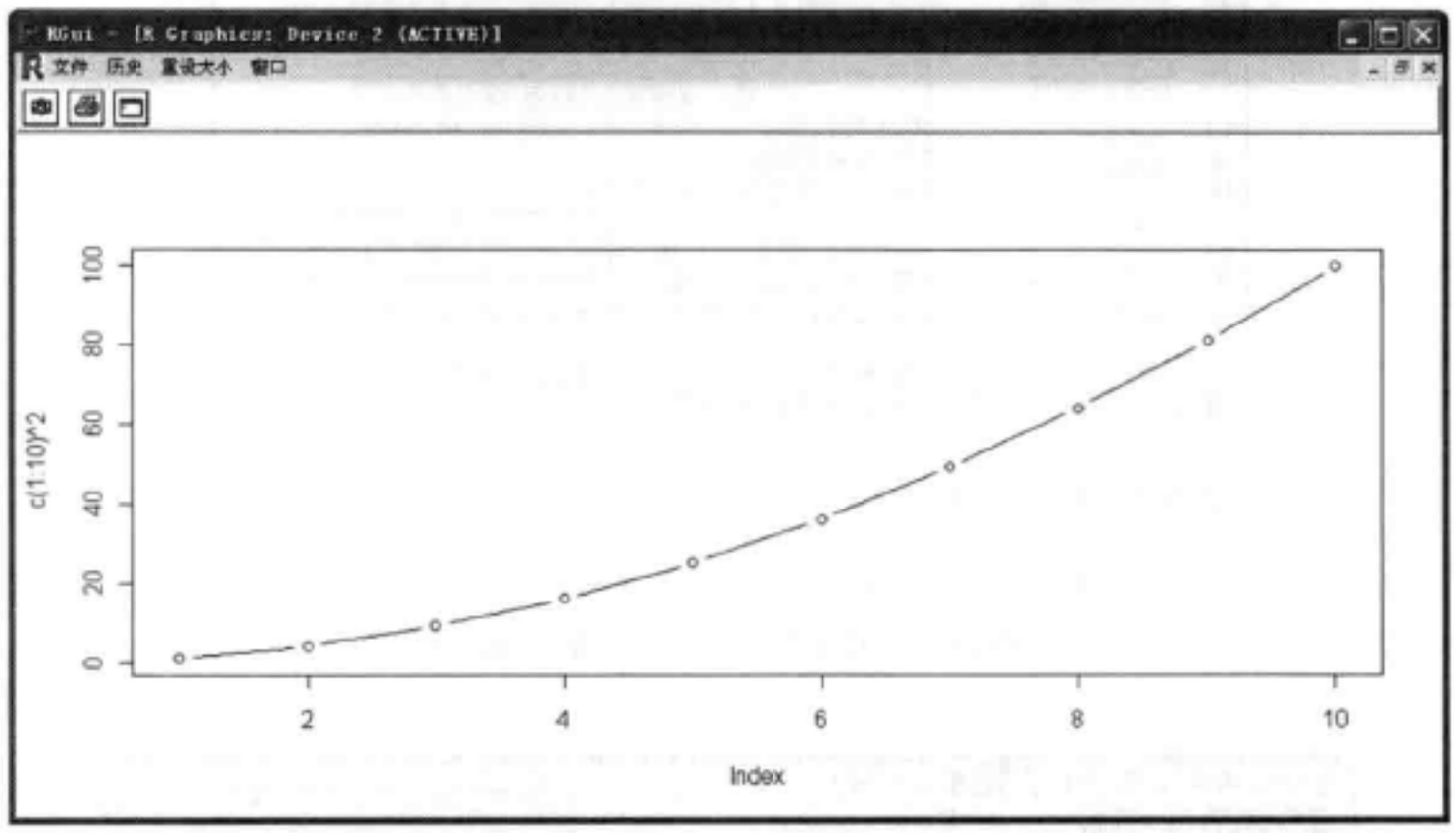


图2-6 绘图窗口

在保存绘图结果时，可通过执行“文件→另存为 →＜格式＞”命令将结果转存为pdf、bmp、jpeg等多种格式。

2.2.3 工作目录

启动R软件后，会在本地硬盘中为工作目录设置一个路径，通过该路径即可寻找需要的数据和文件。默认情况下，工作目录是R软件的安装路径。通过getwd()命令可以读取工作目录的路径，其格式如下：

```
getwd()
  [1] "d:/My Documents"
```

可以看到，读取后的返回值是字符串形式的，也可以使用如下setwd()命令来设置工作目录。

```
setwd(paste(getwd(),"/r",sep=""))
getwd()
[1]"d:/My Documents/r"
```

其中，paste是字符串合并函数，setwd（paste（getwd(),"/r",sep=""））命令返回的字符串为"d:/My Documents/r"。sep是合并连接符设置参数，默认为空格。此处设置为无任何连接符。

不过要注意，R语言采用的是UNIX路径规则，对于Windows系统的用户来说，如果在路径中使用了Windows常用的“/”作为目录层级间隔符，则会出错。例如：

```
setwd("d:\My Documents\r")
```

```
错误于 setwd("d:My Documents\r") 无法改变工作目录。
此外，还会出现 Warning messages，它表示：
1) 不认识字符串里 '\M' 这样的逸出号。
2) 从 "d:\My Documents\r" 里删除了不认识的逸出号。
```

既然R语言继承了UNIX路径规则，就可以使用“.”来代表当前路径，用“..”来代表上级路径，例如：

```
setwd("d:/My Documents")
setwd("./r")
getwd()
    [1]"d:/My Documents/r"
```

当然，也可以在R软件中执行“文件→改变工作目录”命令来对工作目录进行设置。

2.2.4 命令行交互

前面介绍了如何设置工作目录，下面介绍命令行的使用方式。在图2-7中，红色的竖线是命令输入光标，单击后可直接输入R语言命令。

```
> setwd("d:/My Documents")
> setwd("./r")
>
```

图2-7 命令输入

R软件交互模式支持类似于文本复制和粘贴的操作，如图2-8所示。

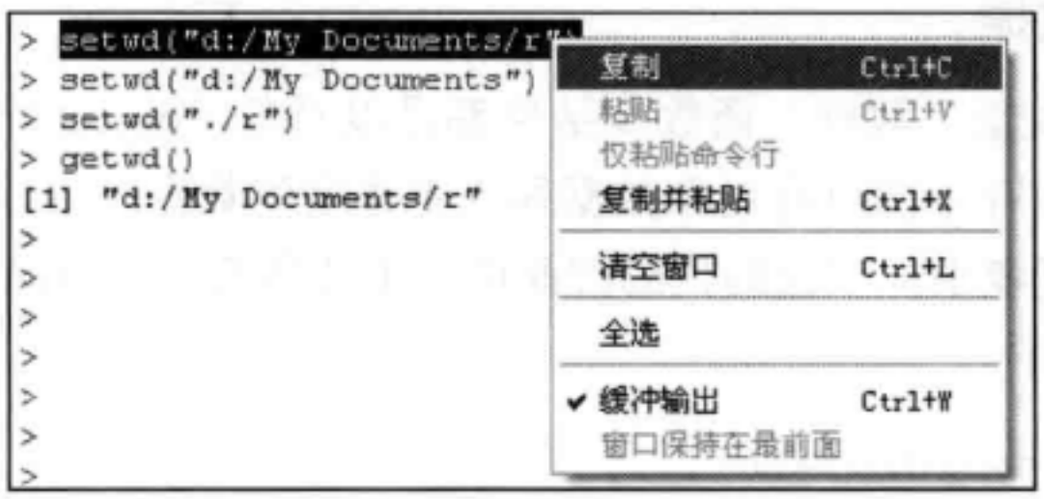


图2-8 复制和粘贴操作

如果想查看之前输入过的历史命令，可按上下方向键。

输入命令后，按Enter键即可显示结果，如图2-9所示。

```
> getwd()
[1] "d:/My Documents/r"
> setwd("d:\My Documents//r")
错误于setwd("d:My Documents//r") : 无法改变工作目录
此外: Warning messages:
1: 不认识字符串里'\M'这样的逸出号
2: 从"d:\My Documents//r"里删除了不认识的逸出号
```

图2-9 命令及输出结果

在图 2-9 中，红色是输入命令，蓝色是命令结果的终端展示。

使用 sink() 命令可以将所有后续的输出由终端转向一个外部文件，例如下：

```
sink("d:/My Documents/r/output.txt")# 把后续代码中的输出写入一个 txt 文件中
sink()# 将把后续代码输出重新恢复到终端上展示
```

2.2.5 命令脚本文件

使用脚本，R 软件可支持批量执行命令模式，具体步骤如下：

1）把需要批量执行的命令写入一个脚本文件，然后使用 source 命令来批量执行其内部命令，例如：

```
setwd("d:/My Documents/r")
print("初始化工作目录成功")
```

2）把以上命令写入“d:/My Documents/script.r”文件内，然后在命令行中运行该脚本即可。例如：

```
setwd("d:/My Documents")
source("d:/My Documents/script.r")
   [1] "初始化工作目录成功！"
getwd()
   [1] "d:/My Documents/r"
```

2.2.6 工作空间数据

运行 R 时，所有变量、数据、函数及结果都是以对象（object）的形式存在于计算机的活动内存中的，此时内存中的所有对象数据称为工作空间数据。可以用 save.image() 函数把工作空间数据转存于硬盘中。下面的代码把此前产生的所有工作空间数据存储在工作目录下 r 目录的 *.RData 文件中。

```
save.image("./r/data.RData")
```

如果想读取硬盘中的 *.RData 文件数据，可以使用 load() 函数。

```
rm(list=ls())      # 删除工作空间内的所有对象数据
a<-c(1:20)         # 建立 a 对象，并赋值一个向量
save.image("./r/data.RData")    # 把工作空间数据存放于硬盘中
rm(a)              # 删除工作空间中的对象 a
a                  # 对象 a 已经不存在了
  错误：找不到对象 'a'
load("./r/data.RData")     # 在硬盘中读取 *.RData 文件
a                          # 对象 a 又重新载入了工作空间中（内存中）
  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

一般在处理大数据挖掘项目时，前期的数据处理最为耗时。此时通常会把项目中有关数据处理的命令单独制成脚本文件（可命名为 init.data.r），并在脚本文件最后使用 save.image() 语句把数据存于硬盘中。命令如下所示：

```
init.data.r 脚本文件的内容
< 数据处理命令 >
save.image(< 目录 /data.RData>)
```

在后续的数据分析脚本（可命名为 sol.data.r）中，可使用 load() 命令先载入数据处理脚本产生的对象数据。这样，在修整分析脚本时，就不必再在数据处理阶段耗费时间了。

```
sol.data.r 脚本文件的内容
load(< 目录 /data.RData>)
< 数据分析命令 >
```

2.2.7 帮助

R 软件提供了丰富的函数帮助系统（不过都是英文的），而且有 HTML 和 CHM 两种形式。在 HTML 帮助文件中，<R 软件安装目录 >\doc\html\index.html 是其帮助文件的存储路径，可以使用 help.start() 命令打开。

在使用帮助文件时，如果想在帮助文件中查找 source 命令，则可以在命令行中直接输入“？ source”来查找，如图 2-10 所示。

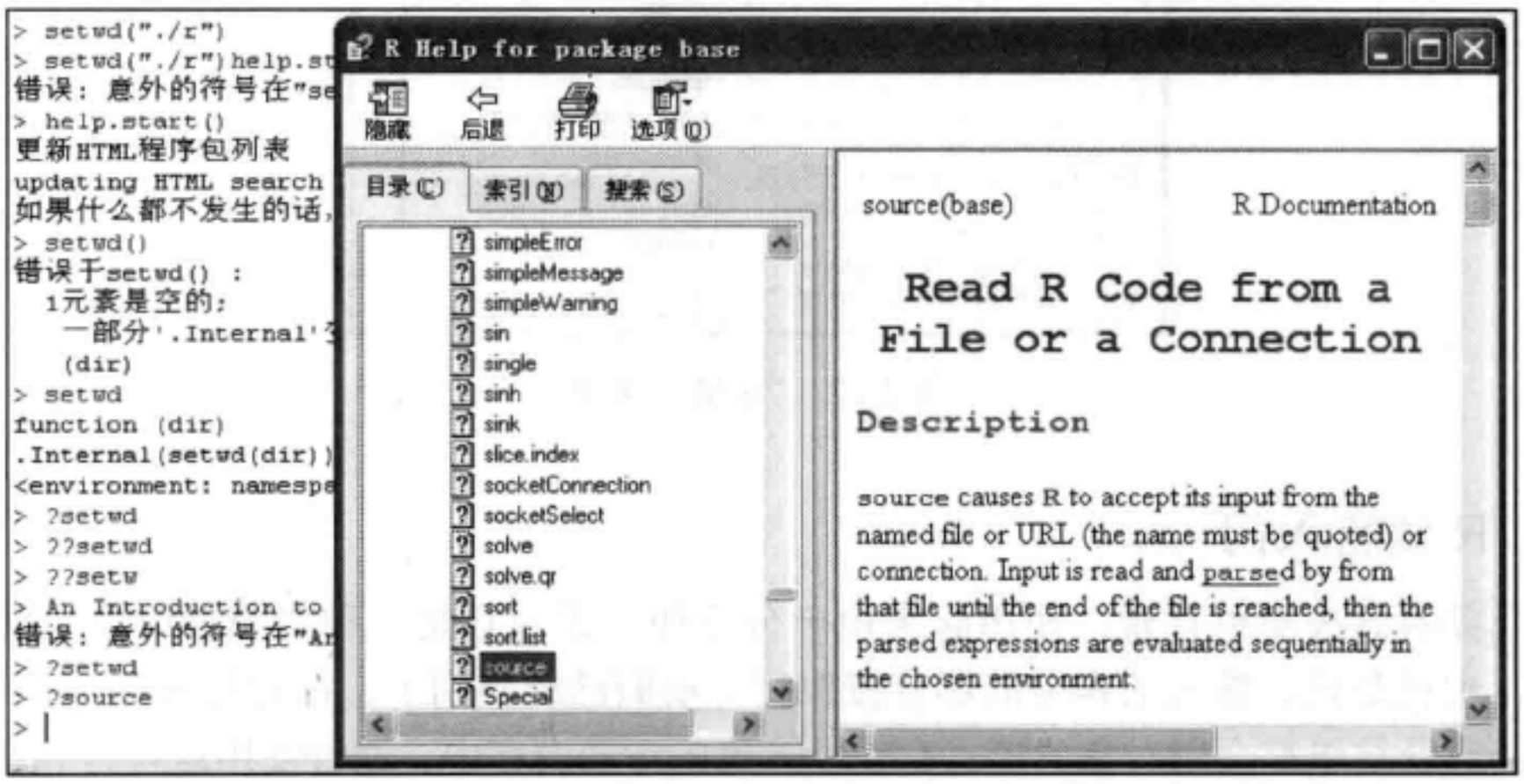


图 2-10 帮助文件的使用

还可以使用“??source”检索所有与 source 相关的信息，如图 2-11 所示。

初学者也可以使用 R 软件中的“帮助”菜单来查找，如图 2-12 所示。

关于“帮助”菜单的使用，这里就不再详细介绍了。

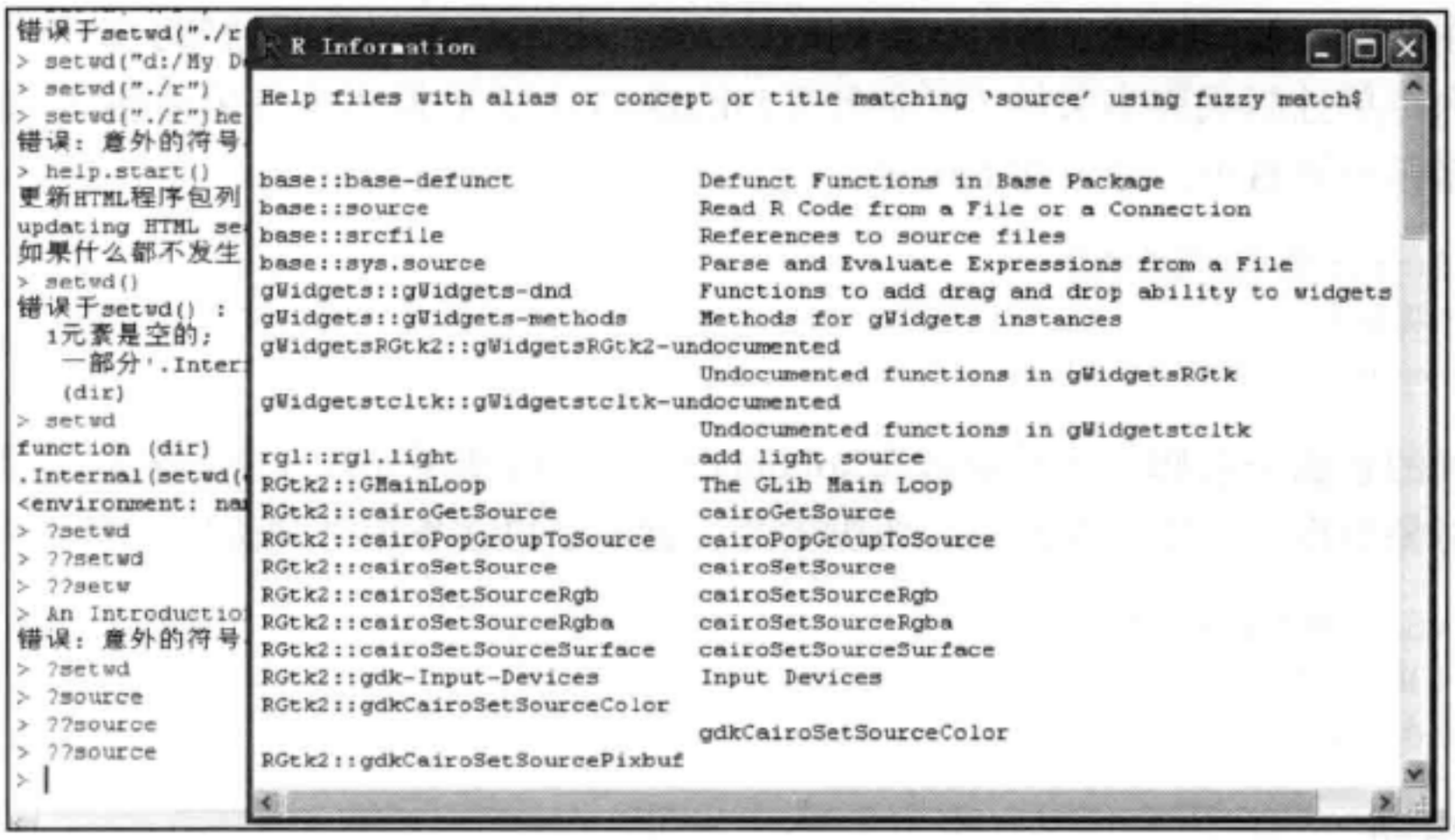


图 2-11 检索所有与 source 相关的信息



图 2-12 “帮助”菜单

2.2.8 R 语言入门

R 语言的语法简单直观，使用起来也十分方便。因为它是一种解释型语言，而不是编译语言，也就是说，输入的命令能够直接执行。前面已经提到了，在运行 R 时，所有变量、数据、函数及结果都会以对象的形式存在计算机的活动内存中，通过使用运算符和函数即可操作这些对象，这也是 R 语言简单好用的原因。

所有可用的 R 函数都存放在一个库（library）中，该库位于 R 软件的安装目录 /library 目录下。该目录包括具有各种功能的包（package），并按照目录的方式组织起来。其中，base 包是 R 的核心包，直接嵌于 R 软件安装包中，它包含了数据读写、操作最基本的函数。在上述目录中的每个包内都有一个子目录 R，并包含了一个同名的文件。例如，在 base 包中，存放函数的文件为：R 软件安装目录 /library/base/R/base。

R对大小写很敏感，即Z和z的含义是不同的，变量名，同一字母的大写和小写分别代表不同的变量对象。另外，R语言名称不能以数字开始。

```
z<-0
例 z
    错误：找不到对象'z'
100a<-0
    错误：意外的符号在"100a"里
```

工作空间可存储变量和函数对象，但是对于求值以及打印命令，并不会保存输出结果。例如，在下面的代码中，工作空间会保存变量z，但是对于z+1的求值和print（z+1）的打印并不会存储于工作空间中。

```
z<-0
z+1
[1] 1
print(z+1)
[1] 1
```

如果R软件监测到输入的命令行未结束，就会给出一个提示符“+”，提示用户可以在第二行和后续行继续输入命令，直到从语法角度来讲命令已经输入完成了为止，如图2-13和图2-14所示。按Esc键可以退出继续输入命令的模式。

```
> if(z>0){a<-1}else{
+
```

图2-13 提示继续输入

```
> if(z>0){a<-1}else{
+ a<-2}
> a
[1] 2
>
```

图2-14 输入完整命令

R语言的注释标识是以井号（#）开始，#后面所跟的都是注释，直到此行结束为止。

注意

初学者常见的一个错误是把正确的英文双引号与括号()错写为中文字符的双引号和括号。下面就是一个把中文格式括号错误写入命令行的例子。

```
> print("正确！")
[1] "正确！"
> print（"正确！"）
错误：意外的输入在"print?>
```

2.2.9 扩展算法包

R软件本身只包含8个基本模块，使用其他函数（如数据分析、文件处理、图形绘制等相关的函数）均要通过安装扩展算法包来完成。

1. 安装包

使用install.packages()可以在命令行模式下安装包，具体步骤如下：

1）在命令行中输入安装包命令。

```
install.packages("qcc")
```

2）选择 CRAN 镜像站点下载 qcc 包，如图 2-15 所示。



图 2-15 CRAN 镜像站点

3）R 软件下载并安装 qcc 包，如图 2-16 所示。

```
+ > install.packages("qcc")
--- 在此連線階段时请选用CRAN的鏡子 ---
警告: 无法在貯藏處http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib/2.9$
试开URL'http://ftp.ctex.org/mirrors/CRAN/bin/windows/contrib/2.9/qcc_2.0.zi$
Content type 'application/zip' length 393697 bytes (384 Kb)
打开了URL
downloaded 384 Kb

程序包'qcc'打开成功，MD5和检查也通过

下载的程序包在
        C:\Documents and Settings\oldlee11\Local Settings\Temp\RtmpKUTo7D\do$
更新HTML程序包的描述内容
>
```

图 2-16 安装 qcc 包

此外，还可以执行“程序包→安装程序包”命令来完成安装。

2. 加载包

使用 library() 或者 require() 可以在命令行模式下将已经安装好的包加载至工作空间内，如图 2-17 所示。

```
> ?qcc
No documentation for 'qcc' in specified packages and libraries:
you could try '??qcc'
> library(qcc)
Package 'qcc', version 2.0
Type 'citation("qcc")' for citing this R package in publications.
> ?qcc
>
```

图 2-17 进入 qcc 函数帮助文档示意图

从上面的命令行界面可以看到，在未载入 qcc 包时键入“？ qcc”命令，无法打开 qcc

包的帮助文件，当然也无法找到 qcc 函数的帮助说明。使用 library 命令载入 qcc 包后，再输入“？ qcc”命令就不会出现错误提示了，并且可以顺利打开 qcc 函数的帮助文件。

2.3 R 语言的数据类型

大家都知道，从变量声明的角度区分，数据类型可以分为静态类型和动态类型。

在编译期间就确定数据类型的语言，叫静态类型语言。该类语言要求在使用任一变量之前必须先声明其数据类型。例如，Java 和 C 就是静态类型语言。

在运行期间才确定数据类型的语言，叫动态类型语言。对于该类语言，不必事先声明其数据类型，而是通过第一次的赋值来定义其数据类型。R 和 Python 为动态类型语言。

从不同数据类型是否可混合使用的角度来区分，语言又可以分为弱类型语言和强类型语言。

在弱类型语言中，不同数据类型间的数据可以混合使用。例如，在 VBScript 中，可以将字符串 '12' 和整数 3 进行连接，从而得到字符串 '123'。

在强类型语言中则相反，不同数据类型间的数据不可以混合使用，必须通过类型转化函数处理后才可以。R 和 Python 就是强类型语言。

了解了 R 语言的数据类型后，再来看看 R 语言支持的数据类型。通常它支持的数据类型有数值型、字符型、复数型和逻辑型。

数值（numeric）类型的取值是实数，在 R 环境中使用数字来表示。例如：

```
a<-9.111
mode(a)
  [1] "numeric"
```

复数（complex）类型的取值可以扩展到虚数，并使用 < 实部数值 >+i< 虚部数值 > 来表示。例如：

```
a<-100+11i
mode(a)
  [1] "complex"
```

逻辑（logical）类型的取值为 TRUE（也可以简写为 T）和 FALSE（也可以简写为 F）。例如：

```
a<-T
mode(a)
  [1] "logical"
```

字符（character）类型的取值是字符串，在 R 语言中，字符串需要使用英文双引号括起来表示。例如：

```
a<-ttt123
    错误 : 找不到对象 'ttt123'
```

```
a<-"ttt123"
mode(a)
    [1] "character"
```

有两种特殊情况不能使用上述4种数据类型来描述：数据的缺失和数据的未知状态。

NA表示数据集中的某数据遗失。在对具有NA样本的变量进行函数操作时，该NA不会被直接剔除。例如，x<-c（10,20,30,NA,40），取mean（x），则结果为NA。如果函数中有na.rm参数，则可以设置为T，以去除NA的影响。虽然NA表示了数据的遗失，但在工作空间数据中（内存中）是记录该数据的，当使用向量长度读取函数length时，返回的向量长度为5。具体代码及结果如下所示：

```
x<-c(10,20,30,NA,40)
mean(x)
    [1] NA
mean(x,na.rm=T)
    [1] 25
length(x)
    [1] 5
```

NULL表示未知的状态。在对具有NULL样本的变量进行函数操作（mean是计算平均数的函数）时，NULL样本不会被计算。而使用print函数输出x数据时，也把NULL忽略了。最能说明NULL与NA不同的是当读取向量的长度时，NULL并不占据任何工作空间，length（c（10,20,30,NULL,40））的返回值为4。而NA是占据工作空间的，length（c（10,20,30,NULL,40））的返回值为5。具体代码及结果如下所示：

```
x<-c(10,20,30,NULL,40)
mean(x)
    [1] 25
print(x)
    [1] 10 20 30 40
print(paste("x1=",x[1],";x2=",x[2],";x3=",x[3],";x4=",x[4]))
    [1] "x1= 10 ;x2= 20 ;x3= 30 ;x4= 40"
length(x)
    [1] 4
```

有时要判断一个数据是否为NA或NULL，这时可以使用is.na和is.null函数。例如：

```
a<-c(1,2,3,NA,4,NULL,9)
is.na(a)
    [1] FALSE FALSE FALSE TRUE FALSE FALSE
is.null(a)
    [1] FALSE
```

在上面的代码中，is.na函数判断向量a的第四个数据（a[4]，R语言中向量下标从1开始）是NA，返回值为TRUE，其余均不是NA，返回值为FALSE。is.null函数判断数据是否为NULL。注意is.na和is.null的区别，is.null在处理向量时不会分别判断内部数据，而是把向量当成一个数据统一判断，其返回值只有一个FALSE。

2.4 对象及其属性

上面介绍了数据类型，接下来介绍组成R语言的基础性概念——对象。前面曾提到，R语言内部数据变量、函数结果、产生的图像等，都是以对象形式存在的。不过由于R语言并非编程型语言，并且R语言的用户也非软件专业出身，所以本书将会弱化函数和图像的对象概念，更多强调的是数据变量的对象概念。

为了更好地组织和查找变量内的样本元素，R语言给出的常用对象类型有向量（vector）、数组（array）、矩阵（matrix）、数据框（data frame）、列表（list）等，如图2-18所示。其中，数据框和列表的内部元素可以使用不同的数据类型。

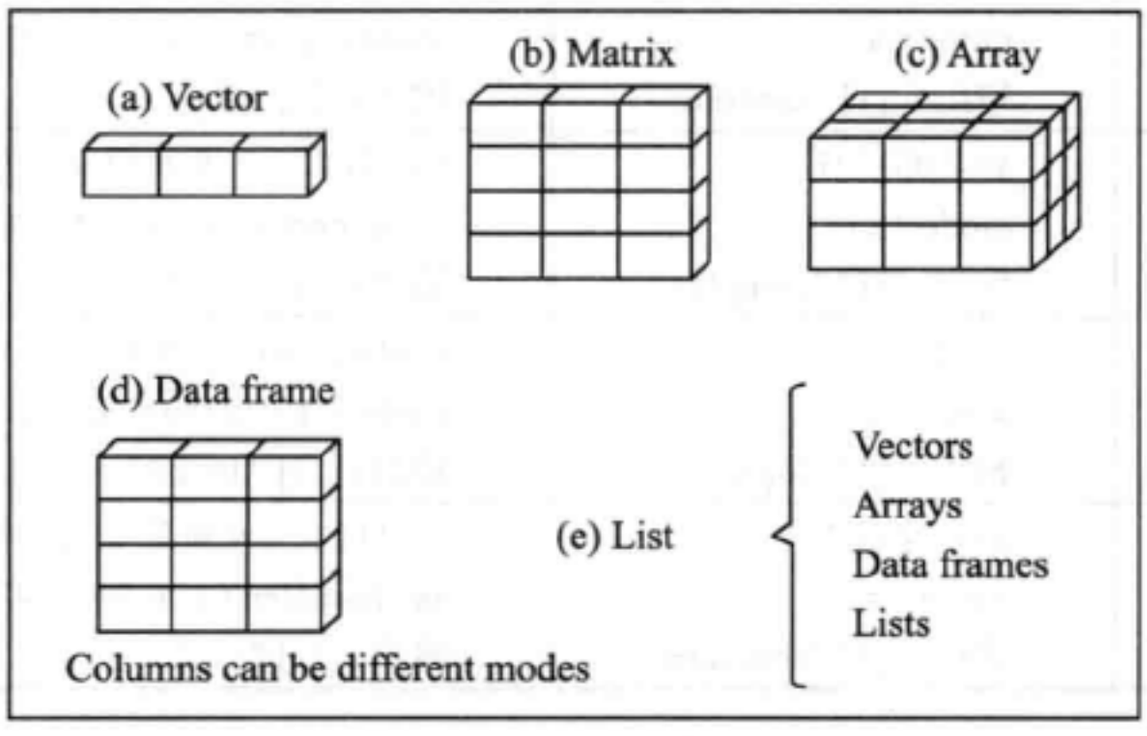


图2-18 R语言常用的数据类型

注意

数据类型和对象类型概念的不同之处。

通俗地说，对象类型是指R语言组织和管理内部元素的不同方式。数据类型则描述了一个变量内元素取值的类型。例如，逻辑类型数据的取值是TURE和FALSE，而数值类型的取值是实数。不同对象类型元素取值的数据类型如表2-1所示。

表2-1 不同对象类型元素取值的数据类型

对象类型	数据类型	是否允许出现不同数据类型
向量	数值型、复数型、字符型、逻辑型	不允许
因子	数值型、复数型、字符型、逻辑型	不允许
数组	数值型、复数型、字符型、逻辑型	不允许
矩阵	数值型、复数型、字符型、逻辑型	不允许
数据框	数值型、复数型、字符型、逻辑型	相同列内元素，其数据类型必须相同；不同列之间的数据类型可以不同
列表	数值型、复数型、字符型、逻辑型	任何元素的数据类型均可不同
时间序列	数值型、复数型、字符型、逻辑型	不允许

2.4.1 固有属性：模式和长度

R语言使用属性来辅助性描述每个对象信息。前面介绍的数据类型就是一个对象最常见的属性，故而也可以把对象内元素的数据类型称作对象的模式。使用mode函数可以读取对象的模式（数据类型），使用as.<数据类型>可以改变对象的模式。

类型的赋值、读取数据类型、转化数据类型3类操作的代码实例如表2-2所示（其中，mode是读取数据类型的函数，as.<数据类型>是修改数据类型的函数）。

表2-2 代码实例

	赋值	读取对象模式：is（对象）	修改对象模式：as.<数据类型>（对象）
数值型	a<-100	a<-100 mode（a） 输出：[1] "numeric"	a<-"100" # 对象a为字符类型 a<-as.numeric（a）# 转化为数值类型 输出：[1] 100
复数型	a<-100+11i	a<-100+11i mode（a） 输出：[1] "complex"	a<-"100" # 对象a为字符类型 a<-as.complex（a）# 转化为复数类型 输出：[1] 100+0i
逻辑型	a<-T	a<-T mode（a） 输出：[1] "logical"	a<-10;b<-0 # 对象a为数值类型 paste（"a=",as.logical（a）,"b=",as.logical（b）） 输出：[1] "a= TRUE b= FALSE"
字符型	a<-" 字符 "	a<-" 字符 " mode（a） 输出：[1] "character"	a<-111 # 对象a为字符类型 as.character（a）# 转化为字符型 输出：[1] "111"

除了数据类型以外，对象的另一个基本属性是对象的长度，前面在讲解NULL和NA时，提到了使用length函数来读取对象的长度属性。值得注意的是，在R中，对象长度可以是0，并可以对超过对象长度的下标赋值，以增加对象的元素，赋值后对象的长度会自动增长至此下标处，而中间未被赋值的元素会被默认赋值为缺失值NA。例如：

```
x<-numeric()
length(x)
    [1] 0
x[2]<-1
x
    [1] NA  1
```

在上面的例子中，使用numeric()为对象x赋值相当于Java语言中的变量声明，不过，由于R语言在不声明对象数据类型的情况下也可以直接赋值使用，所以很少使用。之后的直接赋值x[2]包含了两个意思：

- x变为一个向量类型对象，x的长度为2。
- 未被赋值的x[1]的数值为NA。

2.4.2 读取和设置属性值

除了固有属性length和mode外，要了解其余的属性，可以使用attributes()和attr()函

数来查看，并且可以对其进行修改。这里以一个数据框对象为例进行说明，在建立数据框前，先命名该对象的两个列。代码如下：

```
x<-data.frame(name1=c(1:5),name2=c("一","二","三","四","五"))
attributes(x)
    $names
    [1] "name1" "name2"
    $row.names
    [1] 1 2 3 4 5
    $class
    [1] "data.frame"
attr(x,"names")
    [1] "name1" "name2"
attr(x,"names")[1]<-"new_name1"
attr(x,"names")
    [1] "new_name1" "name2"
```

在将两个列命名为 name1 和 name2 后，使用 attributes() 函数读取对象 x 除了长度和模式以外的所有属性，其中 $names 是列名称属性。然后使用 attr（<对象>，"names"）读取列名称属性，其返回结果为向量，并可以直接赋值，以修改该属性。attr(x,"names")[1]<-"new_name1" 表示把 x 第一列的名称修改为 "new_name1"。对于数据框的详细应用，会在下面的章节中介绍。

2.4.3 对象的搜索和删除

在运行 R 环境时，所有对象均存储在工作空间中，而工作空间又是由计算机内存承载的，因此，一旦某个对象过于庞大，就往往会影响 R 软件环境的运行速度。此时需要查看工作空间内现存的对象列表，并删除其中的某个对象。例如：

```
a<-c(1:10)
ls()
    [1] "a" "data.init1" "data.init2" "data.init3" "data.init4"
    [6] "data.init5" "data2" "data3" "factor" "hc"
    [11] "i" "pattern"
rm(a)
a
    错误 : 找不到对象 'a'
rm(list=ls())
ls()
    character(0)
```

上面的例子使用 ls 函数展示了工作空间中的所有对象，可以看到，其中包括对象 a。然后使用 rm（a）删除对象 a，再查看 a 时会显示“找不到对象”的错误提示。最后利用 rm（list=ls()）删除所有对象。

2.5 向量

向量是以一维数组的方式管理数据的一种对象类型。可以说向量是R语言中最基本的数据类型，很多算法函数都是以向量的形式输入的。

2.5.1 创建向量

R语言使用c（<元素1>,<元素2>…<元素n>）来创建向量，向量内各元素使用逗号分隔。例如：

```
a1<-c(1,2,3,4)
mode(a1)
    [1] "numeric"
a2<-c("一","二","三","四")
mode(a2)
    [1] "character"
a3<-c(T,F,T,F)
mode(a3)
    [1] "logical"
```

上面分别使用数值型、字符型和逻辑型元素组成了向量。而在下面的例子中，向量内的元素是由不同数据类型组成。数值类型+字符类型组成向量时，其模式为字符类型；数值类型+逻辑类型组成向量时，其模式为数值类型；字符类型+逻辑类型组成向量时，其模式为字符类型。

```
a1_2<-c(1,2,"一","二")
mode(a1_2)
    [1] "character"
a1_3<-c(1,2,F,T)
mode(a1_3)
    [1] "numeric"
a2_3<-c("一","二",T,F)
mode(a2_3)
    [1] "character"
```

2.5.2 向量索引

1. [下标]方式索引

R语言与其他类型语言的最大区别就是，下标始于1而非0。可使用**向量对象[下标]**方式读取该下标的元素，并且可**用向量对象[-下标]**方式读取除该下标元素外的其他元素。例如：

```
a<-c("一","二","三","四")
a[1]
    [1] "一"
a[-1]
```

```
    [1] "二" "三" "四"
```

除了使用一个数据作为下标外，还可以使用一个向量，例如：

```
a[c(1,3)]
    [1] "一" "三"
a[-c(1,3)]
    [1] "二" "四"
```

2. which 方式索引

使用 which 函数可以通过向量内元素的数值，来得到此元素的向量下标。如果有多个元素符合筛选要求，则以向量方式返回多个下标。例如：

```
which(a==" 一 ")
    [1] 1
which(a==c(" 一 "," 四 "))
    [1] 1 4
which(a!=" 四 ")
    [1] 1 2 3
which(a==" 五 ")
    integer(0)
```

对于数值类型的向量，不仅可以使用等于和不等于方式来返回向量下标，还可以使用大于或小于等方式来返回。例如：

```
a=c(11,12,13,14)
which(a>12)
    [1] 3 4
which(a<=13)
    [1] 1 2 3
which(a>12 & a<=13)
    [1] 3
which(a>12 | a<=13)
    [1] 1 2 3 4
```

which.max 和 which.min 用于返回数值型向量中最大和最小元素的下标。例如：

```
a=c(11,12,13,14)
which.min(a)
    [1] 1
which(a==min(a))   # 同 which.min(a)
    [1] 1
which.max(a)
    [1] 4
which(a==max(a))   # 同 which.max(a)
    [1] 4
```

通过 which 方式得到下标后，可以轻松索引特定的元素。例如：

```
a=c(11,12,13,14)
```

```
a[which(a>12 & a<=13)]
    [1] 13
```

3. subset 方式索引

subset 函数可以方便地索引向量、矩阵以及数据框，这里先简单介绍向量的索引。上例中的 a[which（a>12 & a<=13）] 也可以使用

```
subset(a,a>12 & a<=13)
    [1] 13
```

来简单代写。

4. match 方式索引

match 可以称作匹配函数，其函数形式为：

```
match(x, table, nomatch = NA_integer_, incomparables = NULL)
```

其中具体参数的含义如表 2-3 所示。

表 2-3 match 函数的参数说明

参数	说明
X	查询的对象
table	匹配的数值，可以是向量形式
nomatch	如果不匹配，返回的数值，默认返回缺失值 NA
incomparables	设置 table 参数中无效的匹配值，默认不设置，即 NULL

下面来看几个实例。

实例 1，使用 match 函数在向量 a 中匹配 11 和 13。

```
a<-c(11,11,12,12,13,13,14,14)
match(a,c(11,13))
    [1]  1  1 NA NA  2  2 NA NA
```

上面的返回值是一个与 a 等长的向量，其中匹配 11 的元素位置返回 1，匹配 12 的元素位置返回 2，均不匹配的返回 NA。

实例 2，设置 nomatch 参数为 0。

```
match(a,c(11,13),nomatch=0)
    [1] 1 1 0 0 2 2 0 0
```

此时均不匹配的返回 0，而不再是默认的 NA。

实例 3，设置 incomparables 参数为 13。

```
match(a,c(11,13),nomatch=0,incomparables=13)
    [1] 1 1 0 0 0 0 0 0
```

这种情况下将不再对 13 进行匹配。

5. %in% 方式索引

%in% 命令可以判定对象是否包含某项数据，如果存在，则返回 T，否则返回 F。例如：

```
a<-c(11,11,12,12,13,13,14,14)
c(11,15)%in%a
    [1]  TRUE FALSE
```

判定向量 a 是否包含 11 和 15。由于 a 包含 11，不包含 15，所以返回值为向量 TRUE 和 FALSE。

2.5.3 向量编辑

如果向量建立好后，还要对向量进行修改，该怎么办？在 R 语言中，修改向量内某元素很简单，只要通过索引找出特定的元素，然后直接使用 <- 进行赋值即可。不过除了改变元素内容外，R 语言还提供了扩展向量和删除元素的功能。

1. 向量扩展

R 语言可对对象的长度进行任意扩展。例如，使用 c（<向量 1>,<向量 2>…<向量 n>）即可将 n 个向量合并，使其长度扩展。例如下面的代码把原有的向量 a 扩展 3 个新元素。

```
a<-c(1,2,3,4)
a<-c(a,c(5,6,7))
a
    [1] 1 2 3 4 5 6 7
```

2. 元素的删除

要删除向量内某一元素，可使用对向量重新赋值的方式来实现。例如，a[-3] 表示向量 a 中除了第三个元素以外的所有元素，把它重新赋予 a 后实现删除 a[3] 的效果。

```
a<-c(1,2,3,4)
a<-a[-3]
a
```

2.5.4 向量排序

在数据处理中，样本的排序很重要。R 语言可以依据数值的大小使用 sort 进行正排序，使用 rev 依据下标进行倒排序。结合 sort 和 rev 使用 rev（sort（<向量>）），可以对向量依据大小进行倒排序。例如：

```
a<-c(1,4,6,2,4,5,2)
rev(a)      #依据下标进行倒排序
    [1] 2 5 4 2 6 4 1
sort(a)     #依据大小进行正排序
    [1] 1 2 2 4 4 5 6
rev(sort(a))        #依据大小进行倒排序
    [1] 6 5 4 4 2 2 1
```

2.5.5 向量去重

有时可能希望删除向量内重复的元素，这时可以使用 unique< 向量对象 > 来实现。例如：

```
a<-c(1,2,1,4,2,5,1,1)
unique(a)
    [1] 1 2 4 5
```

在本例中，对包含重复元素的向量 a 进行了去重操作。此功能同样也可以使用 table 函数来实现，不过比较麻烦。例如：

```
as.numeric(names(table(a)))
```

2.5.6 缺失值处理

实际工作中，数据集很少是完整的，许多情况下样本中都会包括若干缺失值 NA，这在进行数据分析和挖掘时比较麻烦。R 语言通过 na.fail 和 na.omit 函数可以很好地处理样本中的缺失值。这两个函数的说明如表 2-4 所示。

表 2-4 na.fail 和 na.omit 函数说明

函数	说明	
na.fail	na.fail（< 向量 a>）	如果向量 a 内包括至少 1 个 NA，则返回错误； 如果不包括任何 NA，则返回原有向量 a
na.omit	na.omit（< 向量 a>）	返回删除 NA 后的向量 a
	attr（ na.omit（< 向量 a>）,"na.action"）	返回向量 a 中元素为 NA 的下标

下面来看如下相关示例：

```
data<-c(1,2,NA,2,4,2,10,NA,9)
data.na.omit<-na.omit(data)
data.na.omit
    [1]  1  2  2  4  2  10  9
    attr(,"na.action")
    [1] 3 8
    attr(,"class")
    [1] "omit"
attr(data.na.omit,"na.action")
    [1] 3 8
    attr(,"class")
    [1] "omit"
```

其中，函数 na.fail 和 na.omit 不仅可以应用于向量，也可以应用于矩阵和数据框。另外还可以使用 !x 方式方便地删除 NA。例如：

```
a<-c(1,2,3,NA,NA,2,NA,5)
a[!is.na(a)]
    [1] 1 2 3 2 5
```

其中，is.na 用于判断向量内的元素是否为 NA，返回结果应该是：

向量 FALSE FALSE FALSE TRUE TRUE FALSE TRUE FALSE

即 a 内元素为 NA，其对应的下标元素是 TRUE，反之是 FALSE。！ x 是取非逻辑运算符，!is.na（a）表示 a 内元素不为 NA，其对应的下标元素是 TRUE，反之是 FALSE。通过 a[!is.na（a）] 进行索引后，即可取出 a 内不为 NA 的元素，将其过滤。

2.5.7 向量间操作

前面通过大量示例说明了如何编辑和处理一个向量，下面介绍如何操作多个向量。

1. 比较向量大小

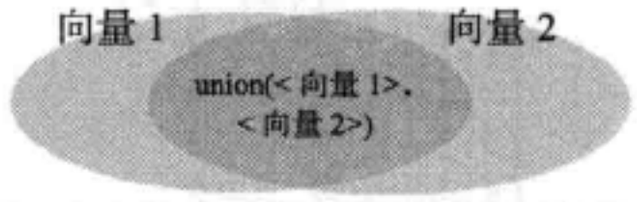
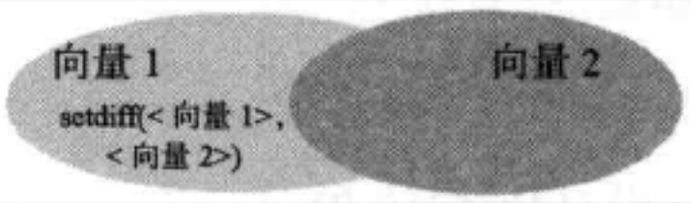
使用 pmin（< 向量 1>,< 向量 2>…< 向量 n>）可以依次比较向量 1 至向量 n 内的各元素，并把较小的元素组成新向量。pmax 则是把较大的元素组成新向量。例如：

```
a<-c(1,2,9,3,5)
b<-c(2,3,5,1,9)
c<-c(2,3,4,5,1)
pmin(a,b,c)
    [1] 1 2 4 1 1
pmax(a,b,c)
    [1] 2 3 9 5 9
```

2. 向量间的交集、并集、补集

除了比较向量间大小外在处理样本间的关系时还有一组更常用的操作，即取交集、并集、补集。这组操作使用的函数如表 2-5 所示。

表 2-5 交集、并集、补集使用的函数

函数	说明	图示
intersect（< 向量 1>,< 向量 2>）	返回由既属于向量 1，也属于向量 2 的元素（交集）所组成的向量	
union（< 向量 1>,< 向量 2>）	返回由属于向量 1 或者属于向量 2 的元素（并集）所组成的向量	
setdiff（< 向量 1>,< 向量 2>）	返回由属于向量 1 而不属于向量 2 的元素（补集）所组成的向量	

2.6 矩阵和数组

向量 vector 用于描述一维数据，是 R 语言中最基础的数据结构形式，然而在很多情况下，数据是以二维甚至多维的形式存在的。利用矩阵 matrix 可以描述二维数据，和向量类似，其内部元素可以是实数、复数、字符、逻辑型数据。矩阵 matrix 使用两个下标来访问元素，A[i,j] 表示矩阵 A 第 i 行、第 j 列的元素。

多维数组 array 可以描述多维数据。array 有一个特征属性叫维数向量（dim 属性），它的长度是多维数组的维数，dim 内的元素则是对应维度的长度。假设多维数组 A 表示是一个四维数据，其中，第一维的长度是 10，第二维的长度是 5，第三维的长度是 8，第四维的长度是 9，那么 A 的维数向量 dim（A）是向量 c（10,5,8,9）。通过维数向量，可以很好地描述多维数据。

2.6.1 创建矩阵

在 R 中，可以使用 matrix 函数并以向量形式输入矩阵中的全部元素，使用 ncol 和 nrow 可设置矩阵的行和列数，从而创建一个矩阵。例如：

```
data<-c(1:10)
a<-matrix(data,ncol=2,nrow=5)
a
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10
```

在上面的示例中，data 是一个向量，是元素为 1 ～ 10 的实数。使用 matrix 函数，把向量 data 转化为 2 列（ncol=2）5 行（nrow=5）的矩阵。注意，向量 1 ～ 10 的填充方式为以列为单位填充。如果想以行为单位填充，则可以把 matrix 函数的 byrow 参数置为 T。例如：

```
a<-matrix(data,ncol=2,nrow=5,byrow = T)
a
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
[4,]    7    8
[5,]    9   10
```

对于 matrix 函数的行数参数 nrow 和列数参数 ncol，可以通过 dim 函数来获取并编辑。例如：

```
data<-c(1:10)
a<-matrix(data)
dim(a)<-c(2,5)
a
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
```

其中，dim（＜矩阵＞）是由矩阵的行数和列数这两个元素组成的向量，我们称其为维数向量。

在创建矩阵时，也可以使用 dimnames 参数设置行和列的名称。例如：

```
a<-matrix(data,ncol=2,nrow=5,dimnames=list(c("r1","r2","r3","r4","r5"),c("c1","c2")))
a
     c1 c2
r1   1  6
r2   2  7
r3   3  8
r4   4  9
r5   5 10
```

dimnames 参数以 list 列表形式进行输入，列表数据的具体说明会在下面章节中讲解。在上面的示例中，list（c（"r1","r2","r3","r4","r5"）,c（"c1","c2"））表示行名称为 r1 ～ r5，列名称为 c1 ～ c2。设置好行和列的名称后，也可以直接通过 dimnames（<矩阵>）来读取并编辑行列名称。

前面建立矩阵的过程中实质是利用了向量内的元素，即通过设置列数 ncol 和行数 nrow 来转化为矩阵。同样，也可以使用 as.vector（<矩阵>）将矩阵反过来转化为向量。例如：

```
data<-c(1:10)
a<-matrix(data,ncol=2,nrow=5)
b<-as.vector(a)
b
 [1]  1  2  3  4  5  6  7  8  9 10
```

其中，col 是列的简称，row 是行的简称。

2.6.2 矩阵索引

由于矩阵 matrix 是用于管理二维数据的，所以使用两个下标便可以表示矩阵内的元素。仍使用上面的例子来说明。

```
a<-matrix(data,ncol=2,nrow=5,dimnames=list(c("r1","r2","r3","r4","r5"),c("c1","c2")))
a
     c1 c2
r1   1  6
r2   2  7
r3   3  8
r4   4  9
r5   5 10
a[3,2]
[1] 8
a["r3","c2"]
[1] 8
a[[8]]
[1] 8
```

1. 使用行列下标来索引

a[3,2] 表示矩阵 a 第 3 行第 2 列的元素 8。

2. 使用行和列名称来索引

对于已经为行和列赋值的矩阵，可以用行和列的名称来代替行列的下标。a["r3","c2"] 和 a[3,2] 代表的是相同的元素。

3. 使用一维下标来索引

我们知道，矩阵a实质是由一维向量c（1:10）组成的（ as.vector（a）为c（1:10）），而矩阵a第3行第2列的元素实质就是向量c（1:10）的第8个元素，因此可以用a[[8]]来表示相应的元素。

以上介绍了矩阵内某一元素的索引，由于二维矩阵可以看作是多个一维向量以行或列的形式组合起来的，所以通过矩阵也可以很方便地找到矩阵中的某行或者列。a[i，]返回矩阵a第i行的所有元素，并以向量形式承载。a[,j]返回矩阵a第j列的所有元素，并以向量形式承载。

```
a[1,]     # 以向量形式返回矩阵 a 第 1 行的所有元素
   [1] 1 6
a[,1]     # 以向量形式返回矩阵 a 第 1 列的所有元素
   [1] 1 2 3 4 5
```

2.6.3 矩阵编辑

1. 矩阵合并：cbind 和 rbind

rbind函数可以以行的形式把多个矩阵或向量合并在一起，扩展成一个新的矩阵。例如：

```
> data1<-c(1:6)
> a1<-matrix(data1,ncol=2,nrow=3)
> a1
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> a1.1<-rbind(a1, c(7,8) )
> a1.1
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
[4,]    7    8
> data2<-c(7:10)
> a2<-matrix(data2,ncol=2,nrow=2)
> a2
     [,1] [,2]
[1,]    7    9
[2,]    8   10
> a1.2<-rbind(a1,a2)
> a1.2
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
[4,]    7    9
[5,]    8   10
```

使用 rbind 扩展了矩阵的行，同理，可使用 cbind 扩展矩阵的列。

2. 删除矩阵

删除矩阵内某行或某列的方式类似于向量，实质是对矩阵进行重新赋值，只要所赋的值中不包含欲删除的行或列的元素即可。例如：

```
data<-c(1:10)
a<-matrix(data,ncol=5,nrow=2)
a
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
a<-a[,-1]
a
     [,1] [,2] [,3] [,4]
[1,]    3    5    7    9
[2,]    4    6    8   10
```

其中，a[,−1] 表示矩阵 a 中除第 1 列外的所有元素，对 a 重新赋值后便删除了 a 中的第 1 列。

同理，a<−a[−1,] 删除了第一行。

2.6.4 矩阵的运算

以如下数据来说明矩阵的运算。

矩阵 A

$$\begin{bmatrix} 1 & 5 & 9 \\ 2 & 6 & 10 \\ 3 & 7 & 11 \\ 4 & 8 & 12 \end{bmatrix}$$

```
A<-matrix(c(1:12),ncol=3,nrow=4)
```

矩阵 B

$$\begin{bmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \\ 13 & 14 & 15 \end{bmatrix}$$

```
B<-matrix(c(4:15),ncol=3,nrow=4,byrow=T)
```

矩阵 C

$$\begin{bmatrix} 12 & 9 & 6 & 3 \\ 11 & 8 & 5 & 2 \\ 10 & 7 & 4 & 1 \end{bmatrix}$$

```
C<-matrix(c(12:1),ncol=4,nrow=3)
```

矩阵 D

$$\begin{bmatrix} -0.6063875 & 1.9445520 & 0.91210545 & -0.4098784 \\ -0.3031623 & 1.4631029 & 0.26384950 & -2.0208419 \\ -0.0526918 & 0.6980914 & -0.03607813 & 0.1713608 \\ -0.1490437 & -0.4592105 & 0.74316101 & -0.5906282 \end{bmatrix}$$

```
D<-matrix(rnorm(16),4,4)
```

1. 一般运算

1）矩阵的加法（A+B）：aij+bij=（A+B）ij。例如：

```
A+B
     [,1] [,2] [,3]
[1,]    5   10   15
[2,]    9   14   19
[3,]   13   18   23
[4,]   17   22   27
```

A+1 #矩阵各个元素均与一个常数相加

```
     [,1] [,2] [,3]
[1,]    2    6   10
[2,]    3    7   11
[3,]    4    8   12
[4,]    5    9   13
```

2）矩阵的减法（A−B）：aij−bij=（A−B）ij。例如：

```
A-B
     [,1] [,2] [,3]
[1,]   -3    0    3
[2,]   -5   -2    1
[3,]   -7   -4   -1
[4,]   -9   -6   -3
```

3）矩阵间各元素分别相乘：aij*bij。例如：

```
A*B
     [,1] [,2] [,3]
[1,]    4   25   54
[2,]   14   48   90
[3,]   30   77  132
[4,]   52  112  180
```

4）矩阵相乘 AB：$\sum_{r=1}^{n} a_{ir}b_{rj}$ =（AB）ij。例如：

```
A%*%C
     [,1] [,2] [,3] [,4]
[1,]  157  112   67   22
[2,]  190  136   82   28
[3,]  223  160   97   34
[4,]  256  184  112   40
```

2. 转置矩阵

aji=（A 的转置矩阵）ij。在 R 语言中，使用 t 函数可以实现矩阵的转置。例如：

```
t(A)
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
```

```
[2,]    5    6    7    8
[3,]    9   10   11   12
```

3. 矩阵求解 solve()

求解 DX=A 中的 X，可以使用如下代码。

```
solve(D,A)
           [,1]       [,2]       [,3]
[1,] 79.0427369 161.607729 244.172721
[2,] 11.8608204  25.510506  39.160192
[3,] 28.0930137  58.009869  87.926725
[4,] -0.5922526  -1.169314  -1.746375
```

4. 矩阵的特征值和特征向量

假设 A 是 n 阶方阵，如果存在数值 m 和非零 n 维列向量 x，使得 Ax=mx 成立，则称 m 是 A 的一个特征值（characteristic value）。非零 n 维列向量 x 则称为矩阵 A 属于（对应于）特征值 m 的特征向量（characteristic vector），简称 A 的特征向量或 A。通过 eigen 函数可以直接求解方阵 D 的特征值和特征向量。例如：

```
d.eigen<-eigen(D)
d.eigen$values
    [1]  1.5493372 -1.2518246 -0.3614882  0.2939847
d.eigen$vectors
           [,1]       [,2]        [,3]       [,4]
[1,]  0.7109595  0.4619719  -0.97858567  0.88671152
[2,]  0.6489598 -0.4688353  -0.01868542  0.23850967
[3,]  0.2511445  0.3807723  -0.17477812  0.39204972
[4,] -0.1015588 -0.6494520   0.10711471  0.05615017
```

其中，eigen（<方阵>）$values 为方阵的特征值，eigen（<方阵>）$vectors 为方阵的特征向量。在上例中，方阵 D 的特征值有 4 个，第 i 个特征值 d.eigen$values[i] 对应的特征向量为 d.eigen$vectors[,i]。

2.6.5 数组

1. 数组创建

了解了矩阵，相信也可以很快地学好多维数组，因为矩阵可以看成是一个特殊的多维数组（维数为 2）。在数组中，也有维数向量概念，只是其长度不再只是为 2 了。可使用 array 函数并设置维数向量来创建一个多维数组。例如：

```
data<-array(c(1:30),dim=c(2,5,3))
data
    , , 1
         [,1] [,2] [,3] [,4] [,5]
    [1,]    1    3    5    7    9
    [2,]    2    4    6    8   10
```

```
, , 2
     [,1] [,2] [,3] [,4] [,5]
[1,]   11   13   15   17   19
[2,]   12   14   16   18   20
, , 3
     [,1] [,2] [,3] [,4] [,5]
[1,]   21   23   25   27   29
[2,]   22   24   26   28   30
```

上面表示建立一个三维数据的数组，其维度是2×5×3。在结果中会依次展示3个2行5列的矩阵。dim函数同样可以用于多维数组中，具体使用方法可以参考前面章节中矩阵的演示。

2. 数组索引

由于数组是用于描述多维数据的，所以数组的索引需要用到多个下标。在上例中，data[1,2,3]对第3个2×5矩阵中第1行第2列元素（23）进行了索引。data[,3,]则表示由第二维度下标是3的全部元素组成的二维数组。

2.6.6 apply函数应用

apply函数可以读取多维数组中某个维度的所有数据并应用其他函数进行数据处理。apply函数的基本形式为：

```
apply(X, MARGIN, FUN)
```

其中具体参数如表2-6所示。

表2-6 apply函数的参数

参数	说明
X	多维数组array
MARGIN	预处理的维数
FUN	多维数组中某一维度元素的处理函数

这里仍些使用上文中的数据来进行说明。

```
data<-array(c(1:30),dim=c(2,5,3));data
, , 1
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
```

apply(data,3,sum)[1]

```
, , 2
     [,1] [,2] [,3] [,4] [,5]
[1,]   11   13   15   17   19
[2,]   12   14   16   18   20
```

apply(data,3,sum)[2]

```
, , 3
     [,1] [,2] [,3] [,4] [,5]
[1,]   21   23   25   27   29
[2,]   22   24   26   28   30
```

apply(data,3,sum)[3]

上例中的data数据是一个三维数据，其中第三维的下标是1～3，apply（data,3,sum）[1]用于计算多维数组data中第三维下标为1的所有元素的总和（fun参数为sum取和函数）。当然，该功能也可以通过sum（data[,,1]）来实现。

2.7 列表和数据框

列表 list 和数据框 data.frame 也是一个二维数据，其中向量 vector、多维数组 array 以及矩阵 matrix 存储的元素，其数据类型是唯一的。列表和数据框内每列元素的数据类型可以不同，列表内每列的长度也可以不同。一般，在使用 R 语言进行数据分析和挖掘的过程中，向量和数据框的使用频率是最高的，list 则在存储较复杂的数据时作为数据对象类型。

2.7.1 列表的创建和索引

list() 函数可用于创建列表对象。例如：

```
data<-list(a=c(1,2,3),b=c("one","tow"),c=T,d=c(3i+4,9i-9));data
$a
[1] 1 2 3
$b
[1] "one" "tow"
$c
[1] TRUE
$d
[1]  4+3i -9+9i
```

在上面的示例中，data 是一个 list 对象，内部元素分别存储于名称为“a”～“d”的 4 列中。每列内元素的数据类型可以各不相同，而且各列内元素的长度也可以各不同，但是相同列中的元素数据类型相同。对于向量的索引，既可以直接使用列下标：<list 对象 >[[下标]] 的形式，也可以使用列名称：<list 对象 >[[" 列名称 "]] 或 <list 对象 >$ 列名称的形式，如表 2-7 所示。

表 2-7 列索引说明

列名称	列内元素的数据类型	列的长度	列的索引		返回各列的数据
			下标	列名称	
a	实数型	3	data[[1]]	data$a 和 data[["a"]]	向量 c（1，2，3）
b	字符型	2	data[[2]]	data$b 和 data[["b"]]	向量 c（"one" ,"tow"）
c	逻辑型	1	data[[3]]	data$c 和 data[["c"]]	T
d	复数型	2	data[[4]]	data$d 和 data[["d"]]	向量 c（4+3i，9+9i）

列表内的元素不仅可以是不同数据类型的向量或数据，也可以是一个列表，即可以使用列表的嵌套来建立更复杂的数据。例如：

```
data<-list(list("one",2,T),c(1:6),c(T,F,F,F))
data
    [[1]]
    [[1]][[1]]
    [1] "one"
    [[1]][[2]]
    [1] 2
```

```
[[1]][[3]]
[1] TRUE

[[2]]
[1] 1 2 3 4 5 6

[[3]]
[1]  TRUE FALSE FALSE FALSE
```

在上例中，list对象data的第一个元素也是一个list。通过这种方式，可以把各个不同数据类型的元素存储于一个list对象中。

```
data[[1]]
    [[1]]
    [1] "one"
    [[2]]
    [1] 2
    [[3]]
    [1] TRUE
```

2.7.2 列表编辑

1. 列表合并

可以使用c（<list对象>,< list对象>）合并和扩展list。注意，list（<list对象>,< list对象>）不能进行list合并，它是实现list的嵌套。例如：

```
data<-c(data,list(e=c(T,T,F,F)))      # 或使用 data<-c(data,e=list(c(T,T,F,F)))
data
    $a
    [1] 1 2
    $b
    [1] "one" "tow"
    $e
    [1]  TRUE  TRUE FALSE FALSE
```

上例在原有list对象data的基础上，使用c()函数扩展了一个名为e的列。

2. 列表转化为向量

unlist函数可以把列表对象转化为向量对象，转化后元素的数据类型被统一。例如：

```
unlist(list(c(1:3),c("one","tow")))
    [1] "1"   "2"   "3"   "one" "tow"
```

2.7.3 数据框的创建和名称

笔者认为，数据框是仅次于向量的最重要的数据对象类型。在R语言中，很多数据分析算法函数的输入都是数据框对象。而且，在使用读取excel/txt等格式数据集的函数时，也是以数据框对象输出的。类似于list，数据框也可以由不同的向量作为列来合成，并且不同

列间的元素可以是不同的数据类型。但是数据框并没有 list 那么灵活，数据框内每个列的长度必须相同。

在实际操作中，通常会用数据框的一列代表某一变量属性的所有取值，用一行代表某一样本数据。

1. 数据框的创建

data.frame 函数可以直接把多个向量建立为一个数据框，并为列设置名称。例如：

```
web.data<-data.frame(date=c("2012.1.1","2012.1.2","2012.1.3","2012.1.4","2012.1.5",
  "2012.1.6","2012.1.7"),pv=c(10234,20119,9087,11093,10099,15899,9908),
  times=c(30.98,33.09,20.09,22.98,17.00,40.09,38.34),
 bounce=c(0.61,0.623,0.679,0.709,0.611,0.54,0.609));web.data
      date    pv    times  bounce
1  2012.1.1  10234  30.98  0.610
2  2012.1.2  20119  33.09  0.623
3  2012.1.3  9087   20.09  0.679
4  2012.1.4  11093  22.98  0.709
5  2012.1.5  10099  17.00  0.611
6  2012.1.6  15899  40.09  0.540
7  2012.1.7  9908   38.34  0.609
```

在上例中，数据框包括 3 列（名称为 pv、times、bounce）7 行数据元素。实际上这是某网站 7 天里的页面浏览量、网站停留时间、网站跳出率的基本数据。

除了使用向量组成数据框外，还可以通过 data.frame 函数把矩阵 matrix 转化为数据框。例如：

```
data.matrix<-matrix(1:12,c(3,4))
data.frame(data.matrix)
  X1  X2   X3   X4
1  1   4    7    10
2  2   5    8    11
3  3   6    9    12
```

2. 数据框的列名称

上面在建立数据框时直接设置了列名称，其实，也可以通过 names（<数据框>）来读取并编辑列名称。在下例中，数据框的第一列名称 date 被改为 new.date。

```
names(web.data)
    [1] "date"   "pv"     "times"  "bounce"
names(web.data)[1]<-"new.date"
names(web.data)
    [1] "new.date"   "pv"     "times"   "bounce"
```

2.7.4 数据框索引

数据框的索引和矩阵类似，由于都是二维数据，所以它也有两个维度的下标，同时数据框的列名称也可以方便地索引数据框的列数据。

1. 索引列

表2-8中给出了列索引的相关信息。

表2-8 列名称和列下标索引示例

	列名称索引	列下标索引
语法	<数据框对象>$列名称 <数据框对象>[[" 列名称 "]]	<数据框对象>[[列下标]] <数据框对象>[, 列下标]
返回对象类型	向量	

例如：

```
web.data$pv
    [1] 10234 20119  9087 11093 10099 15899  9908
web.data[["pv"]]
    [1] 10234 20119  9087 11093 10099 15899  9908
web.data[[2]]
    [1] 10234 20119  9087 11093 10099 15899  9908
web.data[,2]
    [1] 10234 20119  9087 11093 10099 15899  9908
```

如果想读取多列的数据，则可以以<数据框对象>[,列下标向量]方式来获取元素，并以数据框对象的形式返回。注意，不可以采用<数据框对象>[[列下标向量]]形式。例如：

```
web.data[,1:2]
      new.date    pv
1   2012.1.1   10234
2   2012.1.2   20119
3   2012.1.3   9087
4   2012.1.4   11093
5   2012.1.5   10099
6   2012.1.6   15899
7   2012.1.7   9908
```

2. 索引行

通过<数据框对象>[行下标,]，可以直接获取相应行的所有元素，并以数据框对象的形式返回。例如：

```
web.data[1:2,]
      new.date    pv   times  bounce
1   2012.1.1  10234  30.98  0.610
2   2012.1.2  20119  33.09  0.623
```

如果想返回向量形式的数据，则要先把数据框转化为矩阵再进行索引。例如：

```
as.matrix(web.data)[1:2,]
     date         pv      times   bounce
[1,] "2012.1.1" "10234" "30.98" "0.610"
[2,] "2012.1.2" "20119" "33.09" "0.623"
```

3. 索引元素

上面已经介绍了数据框的列索引可以返回一个向量对象，然后再继续索引向量进而索引某一元素。例如，使用 4 种方式来索引 pv 列的第一个元素。

```
web.data$pv[1]
    [1] 10234
web.data[["pv"]][1]
    [1] 10234
web.data[[2]][1]
    [1] 10234
web.data[1,2]
    [1] 10234
```

除了索引一个元素外，还可以索引一块元素。例如，将索引 1（date）;3（times）;2（pv）列和 1 ～ 4 行的数据元素。

```
web.data[1:4,c(1,3,2)]
        date     times    pv
1   2012.1.1   30.98   10234
2   2012.1.2   33.09   20119
3   2012.1.3   20.09   9087
4   2012.1.4   22.98   11093
```

4. subset 函数索引

在 2.5.2 节介绍向量索引时，曾谈到 subset 函数可以方便地索引元素。事实上，该函数同样可以应用于数据框中。例如：

```
subset(web.data,pv<15000&pv>10000)
        date     pv     times  bounce
1   2012.1.1   10234   30.98   0.610
4   2012.1.4   11093   22.98   0.709
5   2012.1.5   10099   17.00   0.611
subset(web.data,pv<15000&pv>10000,times)
        times
1   30.98
4   22.98
5   17.00
```

其中，subset（web.data,pv<15000&pv>10000）索引 web.data 的 pv 列中大于 10000 并小于 15000 的元素所在行的所有元素。subset（web.data, pv<15000&pv>10000, times）索引其中的 times 列元素。

2.7.5 数据框编辑

1. 数据框的合并扩展

在 2.6.3 节中介绍了使用 cbind 和 rbind 函数可以合并扩展矩阵，它们同样也适用于数据框。

1）增加新的样本数据。例如：

```
web.data <-rbind(web.data,list("2012.1.8",10998,39.11,0.677))
```

2）增加数据集的新属性变量。例如：

```
web.data <- cbind(web.data,uv=c(2099,1022,2976,1987,3022,1982,2709))
```

数据框的这些合并函数可以便捷地扩展数据集的样本数据和属性变量。

2. 数据框的删除

同样可以参考2.6.3节中删除的相关内容，数据框删除操作的实质是对数据框进行重新赋值，只要所赋的值内不包含欲删除的行或列的元素即可。例如：

```
web.data <-web.data[,-1] #删除第一列数据
web.data <-web.data[-1,] #删除第一行数据
```

2.7.6 缺失值处理

2.5.6节在介绍处理向量的缺失值的相关内容时，提到了na.omit函数，它同样适用于数据框的缺失值处理。使用na.omit函数可以删除数据框内包括NA的相关行数据。例如：

```
data<-data.frame(a=c(1,2,NA,3),b=c(9:12));data
   a  b
1  1  9
2  2 10
3 NA 11
4  3 12
na.omit(data)
   a  b
1  1  9
2  2 10
4  3 12
```

上例中，直接删除了包括NA的第三行数据。也可以理解为该数据集中第3个样本中的a变量数据是NA，因此直接删除了该样本。

2.8 因子

在研究数据时，一般可以把数据分为分类和数值两种属性类型，如表2-9所示。

表2-9 分类和数值属性类型

属性类型		描述	例子	R语言描述
分类（定性）	标称	标称属性的数值仅是不同的名称，即标称值只提供区分对象的信息（可以进行等于或不等于比较）	用户ID	无序因子 factor
	序数	序数属性的值信息可以确定对象的序（可以进行大于或小于比较）	用户对商品的评分值	有序因子 ordered

（续）

属性类型		描述	例子	R 语言描述
数值（定量）	区间	对于区间属性，值之间的差有意义，即存在测量单位（可以进行加减）	日期	实数
	比率	比率变量，差和比率均有意义（可以进行乘除）	页面浏览量	

以上章节实际介绍的都是数值属性类型的数据，接下来会讲解分类数据的 R 语言实现。

2.8.1 无序和有序因子

1. factor 因子函数

使用 factor 函数可以把向量编码为一个因子，其基本形式为：

```
factor(x, levels = sort(unique(x), na.last = TRUE),labels, exclude = NA,
ordered = FALSE)
```

具体参数的含义如表 2-10 所示。

表 2-10 factor 函数参数说明

参数	说明
x	向量
levels	是因子的水平。如果不指定值，则由向量 x 内不同的值确定
labels	是水平的标签。如果不指定值，则由向量 x 内不同的值所对应的字符串确定
exclude	在转化时，如果想把向量内某些取值的元素转化为缺失值 NA，则设置该参数
ordered	如果取 FALSE（默认取值），则转化后是无序因子； 如果取 TRUE，则转化后是有序因子

2. 使用 as.factor 把一个向量转化为无序因子向量

可以使用 as.factor 把一个向量转化为无序因子向量，例如：

```
a<-as.factor(c(1,2,3));a
    [1] 1 2 3
    Levels: 1 2 3
a[1]<a[2]
    [1] NA
    Warning message:
    In Ops.factor(a[1], a[2]) : < 对因子没有意义
a[1]==a[2]
    [1] FALSE
```

通过上例可知，使用 as.factor 转化的无序因子不能进行大于小于比较，只能进行等于和不等于比较。

3. 使用 as.ordered 把一个向量转化为有序因子向量

前面介绍了将一个向量转化为无序因子向量的方法，那么如何将向量转化为有序因子向量呢？这时可以用到 as.ordered 函数。例如：

```
b<-as.ordered(c(1,2,3));b
    [1] 1 2 3
    Levels: 1 < 2 < 3
b[1]<b[2]
    [1] TRUE
```

通过上例可知，使用 as.ordered 转化的有序因子可以进行大于、小于的比较操作。

4. is.factor 和 is.ordered

is.factor 函数用于判定向量是否为无序因子。is.ordered 函数用于判定向量是否为有序因子。

5. 读取和设置因子水平（levels）

levels（<因子向量>）以向量的形式读取因子水平对应的字符串，同时可以直接编辑修改。例如：

```
a<-as.factor(c(1,2,3)) ; a
    [1] 1 2 3
    Levels: 1 2 3
levels(a)
    [1] "1" "2" "3"
levels(a)[1]<-"one" ; a
    [1] one 2   3
    Levels: one 2 3
```

注意

对于初学者来说，一个常见的错误就是为某一因子元素赋予了水平中没有的数值。例如：

```
> a[1]<-0
Warning message:
In '[<-.factor' ('*tmp*', 1, value = 0) :
  invalid factor level, NAs generated
```

2.8.2 连续数据的离散化

cut 函数可以把数值类型数据依据间隔区间分段，并返回一个因子序列。cut 函数的常用参数如表 2-11 所示。

表 2-11 cut 函数参数说明

参数	说明
X	被分割的向量
labels	分割时依据的间隔区间
include.lowest	分割时的最小间隔区间
ordered_result	如果设置为 T，则返回有大小意义的因子作为结果； 如果设置为 F（默认），则返回无大小意义的因子作为结果

例如，某网站 21 名访问者一天内停留的时间数据如表 2-12 所示。

表 2-12 数据集

用户 ID	网站停留时间（秒）	用户 ID	网站停留时间（秒）	用户 ID	网站停留时间（秒）
1	10	2	12	3	14
4	11	5	3	6	4
7	5	8	189	9	233
10	10	11	13	12	12
13	520	14	67	15	88
16	902	17	7	18	15
19	22	20	59	21	1998

代码如下：

```
time<-c(10,12,14,11,3,4,5,189,233,10,13,12,520,67,88,902,7,15,22,59,981)
time.cut<-cut(time,breaks=c(0,10,30,60,300,1500))
time.cut
  [1] (0,10]   (10,30]   (10,30]   (10,30]   (0,10]   (0,10]   (0,10]
  [8] (60,300]  (60,300]  (0,10]   (10,30]   (10,30]   (300,1.5e+03] (60,300]
  [15] (60,300]  (300,1.5e+03] (0,10]   (10,30]   (10,30]   (30,60]   (300,1.5e+03]
  Levels: (0,10] (10,30] (30,60] (60,300] (300,1.5e+03]
time.cut[1]<time.cut[2]
  [1] NA
  Warning message:
  In Ops.factor(time.cut[1], time.cut[2]) : < 对因子没有意义
```

上例依据分割区间返回了无大小意义的因子向量，在比较 time.cut 的第 1 个和第 2 个元素时，返回了错误提示。

设置 ordered_result 参数为 T 后，返回的因子向量具有了大小意义。

```
time.cut <-cut(time,breaks=c(10,0,60,30,300,1500),ordered_result=T)
time.cut[1]<time.cut[2]
  [1] TRUE
```

通过 labels 参数，还可以设置返回的因子向量的水平值，例如：

```
cut(time,breaks=c(0,10,30,60,300,1500),labels=c(1:5))
  [1] 1 2 2 2 1 1 1 4 4 1 2 2 5 4 4 5 1 2 2 3 5
  Levels: 1 2 3 4 5
```

2.9 字符串操作

在网站分析中，以字符串形式存在的数据很多。例如，在分析用户轨迹时需要分析其点击的链接，而这些链接 URL 就是字符串。因此，网站文本分析的核心数据即为字符串。一个网站使用第三方统计工具提供的页面浏览量、独立用户数、网站停留时间、网站跳出率等

相关数据如下。

```
       date      pv     uv      times     bounce
6   2013.01.14  179079  36264  4分45秒  78.21%
61  2013.01.15  212420  35886  5分0秒   78.07%
62  2013.01.16  198800  38566  4分48秒  80.59%
63  2013.01.17  235485  42786  4分53秒  80.01%
64  2013.01.18  355708  51203  4分57秒  82.35%
65  2013.01.19  339700  61468  4分52秒  75.24%
66  2013.01.20  317080  59060  5分1秒   73.64%
```

通过 Excel 表导出后，times 和 bounce 列均是字符串形式数据，无法直接分析。那该怎么办呢？本节将介绍如何进行字符串操作。

2.9.1 字符串长度 nchar

length（<向量>）可以读取向量内元素的数目，但无法读取一个字符串的实际长度。这时需要用到 nchar 函数，通过该函数可以读取字符串内部字符数目。例如：

```
data<-"4分50秒"
nchar(data)
  [1] 5
```

2.9.2 字符串合并和分割

1. 合并字符串

paste 函数用于合并字符串，其基本形式为 paste（<字符串>…, sep = " "），其中的参数如表 2-13 所示。

表 2-13 paste 函数参数说明

参数	说明
<字符串>…	字符串或者字符串的对象
sep	链接多个字符串时，中间的间隔符默认为空格

示例代码如下：

```
data<-"4分50秒"
data<-paste("网站停留时间:","3小时",data,sep="")
data
  [1] "网站停留时间:3小时4分50秒"
```

2. 分割字符串

strsplit 函数可以依据特定字符串来把字符串分割为列表，其中用于分割的字符串将不再出现。下面将把字符串 "网站停留时间:3 小时 4 分 50 秒" 转化为用户停留时间的实数（以秒为单位）。

```
> data
[1] "网站停留时间:3小时4分50秒"
> data<-strsplit(strsplit(data,split=":")[[1]][2],split="小时") ; data
[[1]]
[1] "3"        "4分50秒"
> data<-unlist(data) ; data
[1] "3"        "4分50秒"
> h<-as.numeric(data[1]) ; h
[1] 3

>min.sec<-data[2] ; min.sec
[1] "4分50秒"
>min.sec<- unlist(strsplit(min.sec,split="分")) ; min.sec
[1] "4"     "50秒"
>min<-as.numeric(min.sec[1])

> sec<-min.sec[2] ; sec
[1] "50秒"
> sec<- unlist(strsplit(sec,split="秒")) ; sec
[1] "50"
> sec<-as.numeric(sec[1])

> times<-3600*h+60*min+sec
> times
[1] 11090
```

提取小时数据

提取分数据

提取秒数据

笔者通常使用 unlist（strsplit（<字符串>,split=<分割符>））函数，并依据分割符来把一个字符串分割为含有多个字符串的向量。

2.9.3 字符串内部字符的读取和替换

1. 使用 substr() 函数进行读取和替换字符串

substr 函数可以直接读取 / 替换字符串中的一段子字符串，其基本形式为：

```
substr(x, start, stop)
```

具体参数的含义如表 2-14 所示。

表 2-14 substr 函数参数说明

参数	说明
x	字符串或者字符串的对象
start	预读取 / 替换字符串的第一个下标
stop	预读取 / 替换字符串的最后一个下标

示例代码如下：

```
a<-"haghfff"
substr(a,2,4)              # 读取字符串 a 的第 2~4 个字符
    [1] "agh"
```

```
substr(a,2,4)<-"kkk"    # 把字符串 a 的第 2~4 个字符替换为字符串 kkk
a
    [1] "hkkkfff"
```

2. 使用 grep () 函数读取列表内的特定字符串

grep（pattern,x）函数可以在字符型列表（参数 x）中找出和特定字符串（参数 pattern）匹配的序列编号。如果不匹配，则返回 integer（0），其长度 length（grep(pattern.x)）=0。其中 pattern 可以是字符串，也可以是一个正则表达式。例如：

```
a<-list("GET /News.htm HTTP/1.0","GET /feed.html HTTP/2.0")
c<-grep(".html",a);c
  [1] 2
```

上例中，字符型列表 a 的第二个字符串包含字符串 ".html"，grep（".html",a）的返回值是 2。如果 a 不是字符型列表，而是一个字符串，则 grep 可用于判别该字符串中是否包含特殊字符，如果包含，则返回 1，不包含，则返回 integer（0）。例如：

```
a<-"GET /News.htm HTTP/1.0"
c<-grep(".html",a);c
  integer(0)
a<-"GET /feed.html HTTP/2.0"
c<-grep(".html",a);c
  [1] 1
```

3. 使用 regexpr()gregexpr() 和函数读取字符串内的特定字符串

regexpr（pattern,text）函数可以在字符串 x 中提取出特定字符串 pattern 的相关信息。其中 pattern 可以是字符串，也可以是一个正则表达式。例如：

```
a<-"GET /News.html HTTP/1.0 feed.html  HTTP/2.0"
b<-regexpr(".html",a);b
  [1] 10
  attr(,"match.length")
  [1] 5
  attr(,"useBytes")
  [1] TRUE
```

通过 regexpr（".html",a），系统返回如下信息：

- 在字符串 a 中，第一个和字符串 ".html" 匹配的起始位置为 10。可以使用 b[1] 来直接读取。
- 在字符串 a 中，第一个和字符串 ".html" 匹配的字符长度为 5。可以使用 attr（b,"match.length"）或者 b[[2]] 来直接读取。
- 在字符串 a 中是否有字符串与 ".html" 匹配，答案为 TRUE。可以使用 attr（b,"useBytes"）来直接读取。

由上例可知，regexpr() 函数只查询匹配第一个特定字符，要想多次匹配需要使用 gregexpr() 函数。例如：

```
a<-"GET /News.html HTTP/1.0 feed.html  HTTP/2.0"
b<-gregexpr(".html",a);b
  [[1]]
  [1] 10 29
  attr(,"match.length")
  [1] 5 5
  attr(,"useBytes")
  [1] TRUE
```

注意

返回的 b 是一个列表，在直接读取起始位置时，可以使用 b[[1]][1] 来表示第一次匹配的起始位置，返回值是 10。使用 b[[1]][2] 表示第 2 次匹配的起始位置，返回值是 29。在直接读取匹配字符串长度时，可以使用 attr（b[[1]],"match.length"）以向量形式返回两次匹配的字符串长度。

4. 使用 chartr () 函数作字符替换

使用 chartr() 函数可以直接进行字符的替换，其查询的规则将以被替换字符串的形式给出，其基本形式为：

```
chartr(old, new, x)
```

表 2-15 给出了具体的参数说明。

表 2-15 chartr 函数参数说明

参数	说明
x	字符串或者字符串的对象
old	预被替换的 x 中的旧字符集合。如果 x 字符串中不包括 old 字符集合，则函数结果仍返回 x 字符串，不做任何处理
new	替换处理后，新补充的字符集合。其长度必须大于等于旧字符集合

示例代码如下：

```
a<-"haghfff"
a<-chartr("k","f",a)        # 字符串 a 中并未包括 k 字符，chartr 对 a 不做任何处理
  [1] "haghfff"
a<-chartr("fff","tt",a)     # 出现了错误：old 字符比 new 字符长了
  错误于 chartr("fff", "tt", a) : 'old' 比 'new' 要长
a<-chartr("hf","01",a)
a
  [1] "0ag0111"
```

在上例中，只有最后一个是正确的，chartr（"hf","01",a）把字符串 a 中的所有 h 字符替换为了 0，所有 f 字符替换为了 1。

注意

不是直接在字符串 a 中截取 hf，并替换为 01。

5. 使用 sub() 和 gsub() 函数进行字符串替换

chartr() 的替换单位是字符，sub() 和 gsub() 函数的替换单位则是字符串。sub 函数基本形式如下：

```
sub(pattern, replacement, x)
```

具体参数的含义如表 2-16 所示。

表 2-16 sub 函数参数说明

参数	说明
pattern	欲被替代的字符串，可以使用正则表达式
replacement	替代后的字符串
x	原始字符串

相应的示例代码如下：

```
a<-"GET /News.html HTTP/1.0 feed.html  HTTP/2.0"
b<-sub(".html","替换",a);b
  [1] "GET /News替换 HTTP/1.0 feed.html  HTTP/2.0"
```

上例把字符串 a 内的字符串 ".html" 部分替换为了字符串 " 替换 "。但是 sub 函数只对第一个 ".html" 进行了替换，要想进行全部替换可以使用 gsub 函数。例如：

```
a<-"GET /News.html HTTP/1.0 feed.html  HTTP/2.0"
b<-gsub(".html","替换",a);b
  [1] "GET /News替换 HTTP/1.0 feed替换  HTTP/2.0"
```

2.9.4 正则表达式

进行字符处理时，往往不是精确地处理某几个字符串，而是处理拥有相似规则的一类字符串，此时就需要使用正则表达式来实现。下面简单介绍正则表达式的语法，以及如何通过 grep、regexpr、substr、sub、gsub 等函数结合正则表达式来处理复杂的字符串。

1. 字符类

字符类用于表达多个字符集合内的任意一个字符，可以使用中括号 [] 来表示字符集合。例如，[aeiou] 可用于匹配任何一个英文元音字母，它既可以代表字符 a、字符 e、字符 i，也可以代表字符 o 或字符 u，但不能代表字符串 "aeiou"。除了在 [] 内逐一列举字符外，还可以指定一个范围。例如，[0-9] 匹配任意一个 0 ～ 9 的数字，[a-z] 匹配任意一个小写英文字母，[A-Z] 匹配任意一个大写英文字母，[a-z0-9A-Z_] 匹配任意一个字母、数字或下划线。

有些特殊字符可以匹配某类字符集合，例如，小数点（.）代表除换行符以外的任意一个字符。

如果想查找一些特殊字符本身，如查找小数点（.）或者逗号（,），就会出现问题：无法指定它们，因为它们会被解释成其他意思。这时就需要使用 \\ 来取消这些字符的特殊意义了。也就是说，使用 \\. 来匹配普通字符（.）。

2. 限定符

在正则表达式中可以使用限定符来指定数量，常用的限定符如表 2-17 所示。

表 2-17 常用限定符

代码 / 语法	说明	代码 / 语法	说明
*	重复零次或更多次	{n}	重复 n 次
+	重复一次或更多次	{n,}	重复 n 次或更多次
?	重复零次或一次	{n,m}	重复 n ～ m 次

例如，http://www.xiami.com/space/u/[0-9]+ 表示 URL：http://www.xiami.com/space/u/ 之后跟着一个或多个数字。当要匹配一般字符 * 时，也需要使用 \\ 把 * 从限定符转化为一般字符。

3. 实例：处理网站日志中的 URL 数据

接下来编写一段代码，用以从网站日志数据的 URL 字段中提取出相应的网页信息。

输入数据：GET /Software.html HTTP/1.0

输出数据：Software.html

```
url<-"GET  /Software.html HTTP/1.0"
real_url<-regexpr("/[a-z0-9A-Z_]+\\.html",url)
page<-substr(url,real_url[1]+1,real_url[1]+attr(real_url,"match.length")-1)
```

正则表达式 /[a-z0-9A-Z_]+\\.html 的含义如表 2-18 所示。

表 2-18 字符串解析说明

序号	正则表达式	意义	url 向量中查询的部分
1	/	字符 /	/
2	[a-z0-9A-Z_]+	由数字、字母或下划线组成的长度大于 0 的字符串	Software
3	\\.html	字符串 .html	.html
4	/[a-z0-9A-Z_]+\\.html	由以上 1-3 项顺序组成的字符串	/Software.html

real_url<-regexpr（"/[a-z0-9A-Z_]+\\.html",url）命令提取了在字符串 url 中与正则表达式匹配的起始位置和字符串的长度信息，再通过 substr 函数可以直接截取出目标字符串。

2.10 常用数据的创建

2.10.1 因子序列的创建

使用 gl 函数可以方便地创建一个因子向量。其基本形式为：

```
gl(n, k, length = n*k, labels = 1:n, ordered = FALSE)
```

具体参数的含义如表 2-19 所示。

表 2-19 gl 函数参数说明

参数	说明	参数	说明
n	因子水平的数目	labels	因子水平的标签
k	重复的次数	ordered	如果为 F（默认），则因子向量是无序因子； 如果为 T，则因子向量是有序因子
length	向量长度，默认为 n × k		

相应的示例代码如下：

```
gl(3,4,labels=c("one","two","three"))
  [1] one   one   one   one   two   two   two   two   three three three three
    Levels: one two three
gl(3,2,length=12,labels=c("one","two","three"))
  [1] one   one   two   two   three three one   one   two   two   three three
    Levels: one two three
```

在上例中，gl（3,4,labels=c（"one","two","three"））可以理解为：产生一个因子水平为"one","two","three" 的无序因子向量，其中每个水平连续重复 4 次。

gl（3,2,12,labels=c（"one","two","three"））则可以理解为：每个水平连续重复了两次产生“one one two two three three”，而后该序列又重复了一次（直至向量长度为 12 为止）。

2.10.2 等差序列的创建

等差数列产生函数 sep 可产生等距间隔的数列，基本形式为：

```
seq(from = 1, to = 1, by = ((to - from) / (length.out - 1)))
```

表 2-20 给出了具体参数的含义。

表 2-20 seq 函数参数说明

参数	说明	参数	说明
from	等差数列的首项数据，默认为 1	by	等差的数值
to	等差数列的尾项数据，默认为 1	length.out	产生向量的长度

相应的示例代码如下：

```
seq(1,-9)  # 只给出首项和尾项数据，by 自动匹配为 1 或 -1
  [1]  1  0 -1 -2 -3 -4 -5 -6 -7 -8 -9
seq(1,-9,length.out=5)    # 给出首项和尾项数据以及长度，自动计算等差
  [1]  1.0 -1.5 -4.0 -6.5 -9.0
seq(1,-9,by=-2)    # 给出首项和尾项数据以及等差，自动计算长度
[1]  1 -1 -3 -5 -7 -9
seq(1,by=2,length.out=10)  # 给出首项和等差以及长度数据，自动计算尾项
  [1]  1  3  5  7  9 11 13 15 17 19
```

从上例可知，通常可以通过首项、尾项、等差、长度中的任意 3 个数据来方便地产生等差数列向量。

2.10.3 随机抽样 sample

在模拟实际数据情况时，常常会使用随机抽样函数来从整体中挑出部分样本数据。随机抽样又分为重复随机抽样和不重复随机抽样两种。重复抽样是指：本次从整体中抽取出的数据样本，在下一次抽取时同样有机会被抽取。不重复抽样就是：一旦被抽取为样本，下次就不能再被抽取了。

sample 函数可以完成随机抽样处理，其基本形式为：

```
sample(x, size, replace = FALSE)
```

具体参数的含义如表 2-21 所示。

表 2-21 sample 函数参数说明

参数	说明
x	整体数据，以向量形式给出
size	抽取样本的数目
replace	如果为 F（默认），则是不重复抽样，此时 size 不能大于 x 的长度； 如果为 T，则是重复抽样，此时 size 允许大于 x 的长度

相应的示例代码如下：

```
sample(c(1:20),size=10)  # 在 1~20 数据中不重复地随机抽取 10 个元素作为样本
 [1]  8  2 20 11 14  6 19  1 12  9
sample(c(1:20),size=30,replace=T)  # 重复地随机抽取 30 个元素作为样本
 [1]  8 20 12 15  8  6 13  9  2 10  7  2 11  8 17  3 16  7 11  9 15 18  9  5
[25] 16  1 18  1 11  2
```

2.10.4 重复序列 rep

rep 是重复序列函数，其基本形式是 rep（x,n）。其中，x 是预重复的序列，可以是任意数据类型的向量或数值，n 是重复的次数。例如：

```
rep(1,9)
 [1] 1 1 1 1 1 1 1 1 1
rep(1:3,3)
 [1] 1 2 3 1 2 3 1 2 3
rep(c("a","b","c"),3)
 [1] "a" "b" "c" "a" "b" "c" "a" "b" "c"
rep(as.factor(c(" 因子 1"," 因子 2"," 因子 3")),3)
 [1] 因子 1 因子 2 因子 3 因子 1 因子 2 因子 3 因子 1 因子 2 因子 3
 Levels: 因子 1 因子 2 因子 3
```

在上例中，rep（1,9）对实数 1 重复了 9 次，产生了新的数列。rep（1:3,3）则对实数向量 1:3 重复了 3 次，组成了新向量 1 2 3 1 2 3 1 2 3。rep（c（"a","b","c"）,3）表示对字符串型向量进行了 3 次重复操作。rep（as.factor（c（" 因子 1"," 因子 2"," 因子 3"））,3）表示对因子序列进行了 3 次重复操作，最终产生的新向量也是一个因子形式的向量。

2.10.5 概率分布

对于一个离散型变量 X，可以直接描述 X 取不同水平时的概率值 $P\{X\}$。例如，可通过对页面访问深度的长期样本观察，把该变量简化为离散变量，并总结出其概率，如表 2-22 所示。

表 2-22 离散变量概率

X（用户每日页面访问深度）	1 页 / 次	2 页 / 次	3 ～ 5 页 / 次	6 ～ 15 页 / 次	16 ～ 30 页 / 次	大于 30/ 页次
$P\{X\}$	0.4309	0.2016	0.1570	0.1343	0.0512	0.025

然而对于不做离散处理的连续型变量 X，无法直接计算出 X 取不同数值时的概率值 $P\{X\}$（它是个无限小的值）。不过借用微积分的思想，可以计算变量 X 小于等于数值 $x0$ 时的概率值 $P\{X<=x0\}$。把点（$x0$，$P\{X<=x0\}$）连续化，进而形成一条曲线，用函数 $F(x)$ 来表达该曲线，它的意义是连续变量 X 小于等于 x 时的概率。我们称该函数 $F(x)$ 是变量 X 的分布函数：$F(x)=P\{X<=x\}$。

对 $F(x)$ 做微分处理，可以得到变量 X 的概率密度函数 $f(x)=\mathrm{d}f(F(x))$。

对 $F(x)$ 取反函数，可以依据概率值 1-$P0$ 来求得此时变量 X 的对应取值 $x0$，即如果 $1-P0=F(x0)$，则 $x0=F^{-1}(1-P0)$。当取 x 为概率时，得到分位数函数 $z(x)=F^{-1}(1-x)$。

下面主要介绍正态分布的 R 语言应用，其他分布的 R 语言应用方式可以类比正态分布。

1. 正态分布 norm

在正态分布中有两个参数：平均值 μ 和方差 σ^2，其方差的平方根 σ 也称为标准差。

正态分布函数 $F(x)=\dfrac{1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{x}\exp\left\{-\dfrac{t-\mu^2}{2\sigma^2}\right\}\mathrm{d}t$。使用 pnorm(x, mean, sd）函数来表达时，其中的参数 x 就是连续变量 x，参数 mean 是平均值 μ，参数 sd 是标准差 σ。其初始默认取值 mean=0,sd=1。

概率密度函数 $f(x)=\dfrac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\dfrac{(x-\mu)^2}{2\sigma^2}\right\}$，并使用 dnorm（x, mean, sd）函数来表达，如图 2-19 所示。

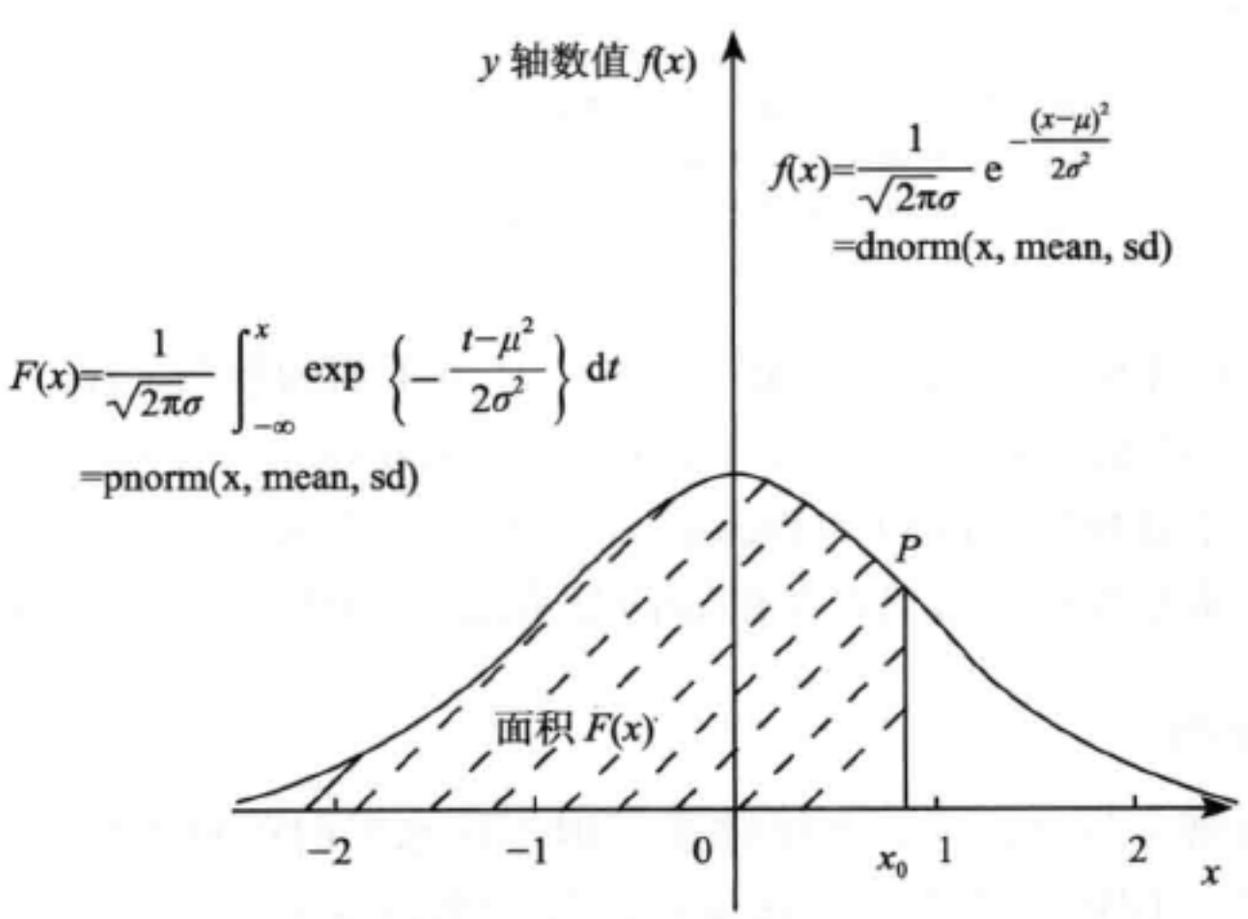


图 2-19 正态分布概率密度曲线说明 1

分位点函数为 $z(x)=F^{-1}(1-x)$，并使用 qnorm（1−x, mean, sd）来表达如图 2-20 所示。其中 x 表示概率密度曲线下的面积，即概率取值。注意 qnorm 表示计算下分位点函数。

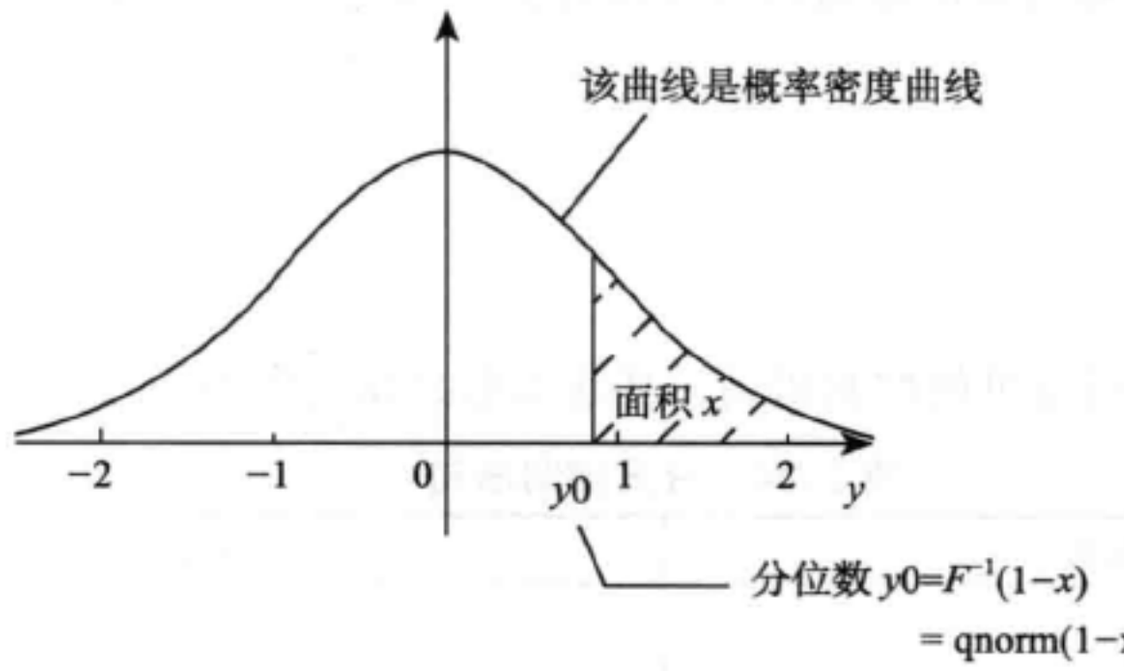


图 2-20 正态分布概率密度曲线说明 2

正态分布随机数的产生函数使用 rnorm（n, mean, sd）来表达，其中 n 是产生的随机数数目。

2. 其他分布

除了正态分布以外，R 语言还给出了其他常用分布函数，如表 2-23 所示。

表 2-23 常用分布函数

分布	在 R 中的名称	参数	分布	在 R 中的名称	参数
beta	beta	shape1, shape2, ncp	log-normal	lnorm	meanlog, sdlog
binomial	binom	size, prob	logistic	logis	location, scale
Cauchy	cauchy	location, scale	negative	binomial	nbinom size, prob
chi-squared	chisq	df, ncp	normal	norm	mean, sd
exponential	exp	rate	Poisson	pois	lambda
F	f	df1, df2, ncp	Student'st	t	df, ncp
gamma	gamma	shape, scale	uniform	unif	min, max
geometric	geom	prob	Weibull	weibull	shape, scale
hypergeometric	hyper	m, n, k	Wilcoxon	wilcox	m, n

在分布名称前添加前缀后可以代表不同的意义，例如：

- p<R 中的分布名称 > 表示该分布的分布函数。例如，pnorm 是正态分布的分布函数。
- d<R 中的分布名称 > 表示该分布的概率密度函数。例如，dnorm 是正态分布的概率密度函数。
- q<R 中的分布名称 > 表示该分布的分位数函数。例如，qnorm 是正态分布的分位数函数。
- r<R 中的分布名称 > 表示该分布的随机序列产生函数。例如，rnorm 是正态分布的随机序列函数。

2.11 控制流

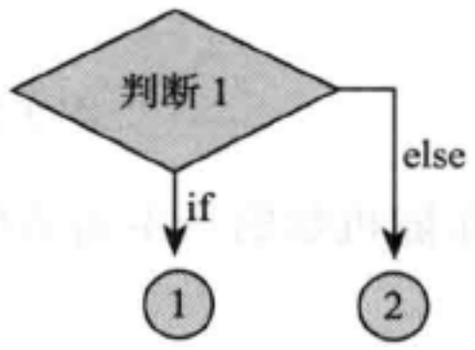
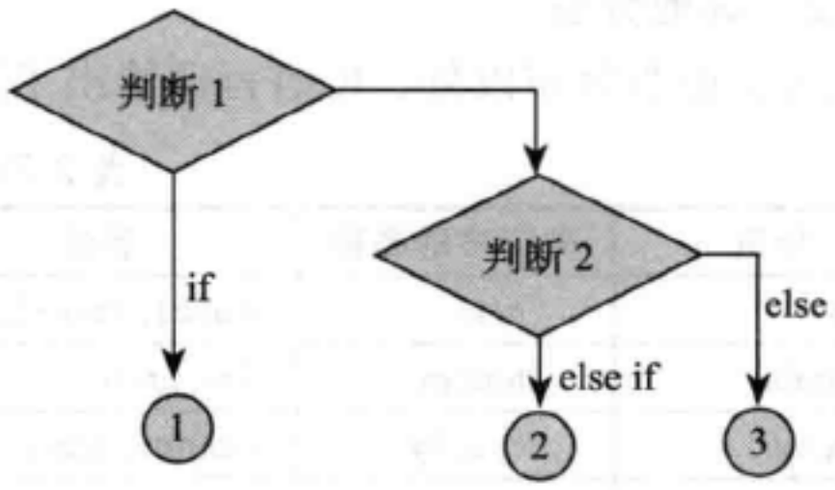
既然R是一种语言，那么就应该包含所有串行语言都有的基本控制语句：分支和循环。下面介绍相应的语句。

2.11.1 分支语句

1. if-else 分支语句

二分支语句 if-else 是最常见的控制语句，其基本形式如表 2-24 所示。

表 2-24 分支控制语句

语句使用形式	流程示意图
if (<判定 1>) { 处理① }else{ 处理② }	
if (<判定 1>) { 处理① }else if (<判定 2>) { 处理② }else{ 处理③ }	

通过 if-else 可以组合成多分支判断语句。如果处理语句中只包含一条语句，则可以省略大括号 {}。例如：

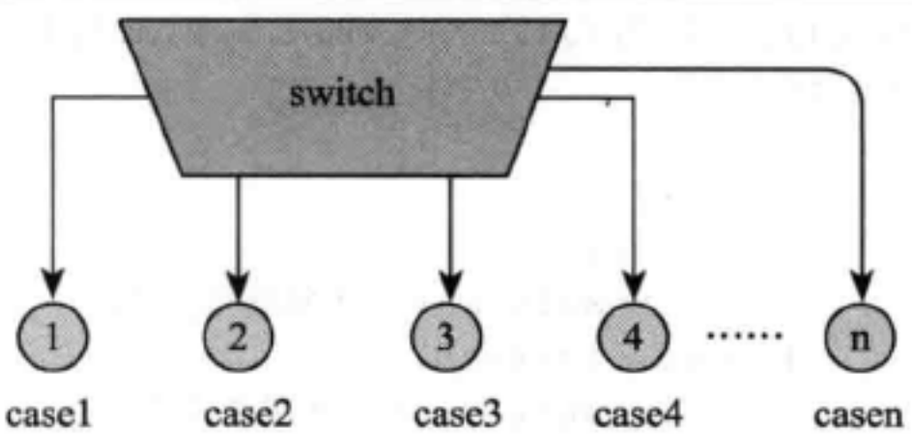
```
if(a>10)
    b<-1          # 处理①中只涉及 1 条语句，可以不必使用大括号 {}
else if (a>5){    # 处理②中涉及 2 条语句，必须使用大括号 {}
    b<-2;c<-2
} else{           # 处理③中涉及 2 条语句，必须使用大括号 {}
    b<-3;c<-3

}
```

2. switch 分支语句

switch 语句可以直接实现多分支语句，但分支时并无判断的优先级区分。其基本形式如表 2-25 所示。

表 2-25 switch 分支语句

语句使用形式	流程示意图
switch（case，case1 处理，case2 处理…casen 处理）	

如果 case 对象等于 n，则执行第 n 条分支的 casen 处理。如果 case 的取值大于 list 对象的长度，则返回值是 NULL。例如：

```
result<-3
switch(result,"低异常点","偏低","正常","偏高","高异常点")
    [1] "正常"
result<-9
switch(result,"低异常点","偏低","正常","偏高","高异常点")
    NULL
```

2.11.2 循环语句

在 R 语言中，可以使用 for、while 以及 repeat-break 实现循环语句。循环语句可以简单依据计数操作（当计数器达到了设定的循环次数时自动停止）或者某一向量来进行轮询。

1. for 循环语句

一组用户在网上的页面浏览量（PV）如表 2-26 所示。

表 2-26 数据集

用户 ID	页面浏览量 PV	用户 ID	页面浏览量 PV	用户 ID	页面浏览量 PV
1	1	6	1	11	1
2	1	7	15	12	2
3	2	8	7	13	3
4	3	9	18	14	1
5	1	10	1	15	1

如果要对用户进行分类，如规定每日页面浏览量 PV 为 1 ～ 5 的是初级用户，6 ～ 15 的为中级用户，PV 在 16 页以上的为高级用户，该如何操作？采用表 2-27 中的两种方式，即可把 PV 数据处理为以用户进行分类的数据。

表 2-27 方式类比

方式一	方式二
pv<-c(1,1,2,3,1,1,15,7,18,1,1,2,3,1,1) result<- "" m<-1 for(i in pv){ if(i<=5){ result[m]<- "初级用户"; } else if(i<=15){ result [m]<- "中级用户"; } else{ result [m]<- "高级用户"; } m<-m+1 }	pv<-c(1,1,2,3,1,1,15,7,18,1,1,2,3,1,1) result<-"" for(i in 1:length(pv)){ if(pv[i]<=5){ result[i]<- "初级用户"; } else if(pv[i]<=15){ result[i]<- "中级用户"; } else{ result[i]<- "高级用户"; } }
不知道循环次数时，可以直接遍历相关向量（i in pv 语句）。	知道循环次数时，可以设置遍历次数（i in 1:length(pv) 语句）

2. while 循环语句

在介绍 while 循环语句时，同样使用上面的数据为例进行说明，如下所示：

```
pv<-c(1,1,2,3,1,1,15,7,18,1,1,2,3,1,1)
i<-1
result<-""
while(i<length(pv)){
    if(pv[i]<=5){
          result[i]<- "初级用户";
    } else if(pv[i]<=15){
          result[i]<- "中级用户";
    } else{
          result[i]<- "高级用户";
    }
    i<-i+1
}
```

由此可以看出，for 实际上是通过遍历一个向量的方式来控制循环次数，while 则是直接设置循环的范围。for 的应用基本可以覆盖 while，所以笔者更加倾向于使用 for 循环语句。

3. repeat-break 循环语句

repeat 是无限循环语句，并且会在达到循环条件后使用 break 语句直接跳出循环。例如：

```
pv<-c(1,1,2,3,1,1,15,7,18,1,1,2,3,1,1)
i<-1
result<-""
repeat{
    if(i>length(pv)){     #设置循环结束时的跳出语句
         break
    }
    if(pv[i]<=5){
         result[i]<- "初级用户";
```

```
    } else if(pv[i]<=15){
          result[i]<- "中级用户";
    } else{
          result[i]<- "高级用户";
    }
    i<-i+1
}
```

实例中的 break 也叫跳出循环命令，执行后将直接跳出循环语句。R 语言还提供了一个 next 语句，执行后只会跳出本次循环，而不会跳出整个循环语句。

2.12 运算符、函数和过程

2.12.1 常用运算符

对一些常用的数值进行处理时，如果仍使用一个函数来实现，则显得过于繁琐。R 语言使用一些符号来便捷地实现对这些数值的处理，这就是运算符。常用的运算符分为算术运算符和逻辑运算符两类。

1. 算术运算符

R 语言中常用的算术运算符如表 2-28 所示。

表 2-28 常用算术运算符

运算符	说明	运算符	说明
+	加	^ 或 **	幂运算
−	减	x%%y	求除法运算后得到的余数。例如，10%%3 的结果为 1
*	乘		
/	除	x% / %y	整数除法。例如，10% / %3 的结果为 3

例如，有些网站的跳出率 bounce 是以小数形式给出的，现在要依据实际情况把其转化为 % 形式。例如，把 0.187905 转化为 18.79%，相关代码如下：

```
bounce<-0.187905
bounce<-((bounce*10000)%/%1)/100 ; bounce
  [1] 18.79
bounce<-paste(bounce,"%",sep="") ; bounce
  [1] "18.79%"
```

2. 逻辑运算符

R 语言中的常用逻辑运算符如表 2-29 所示。

表 2-29 常用逻辑运算符

运算符	说明	运算符	说明
<	小于	!=	不等于
<=	小于等于	!x	非 x

（续）

运算符	说明	运算符	说明
>	大于	x \| y	x 或 y（并集）
>=	大于等于	x & y	x 和 y（交集）
==	等于	isTRUE（x）	测试 x 是否为 TRUE

例如，使用 !x 删除某些缺失数据 NA，代码如下：

```
a<-c(1,2,3,NA,NA,2,NA,5)
a[!is.na(a)]
    [1] 1 2 3 2 5
```

2.12.2 函数

前面的章节介绍了 R 语言系统提供的许多数据处理函数，同时 R 语言还允许用户自己编写函数来创建模型。这可以使算法的可读性更强，后期的维护也更为方便。实际上自定义函数与系统函数没有多大区别。自定义函数的格式如下：

```
func.name<-function(arg_1,arg_2,…) expression
```

其中，func.name 是函数名称，arg_1 和 arg_2…是函数的输入参数，expression 是 R 语言中的表达式。在定义函数时，可以为某些参数直接设置初始值。在定义好函数后，可以使用 func.name（arg_1= arg_1.value,arg_2= arg_2.value,…）来调用函数。如果定义时已经设置好了参数的初始值，则在调用时可以不为参数设置数值。

1. 引例

假设编写一个函数来表达如下数学模型：

$$y=\begin{cases}300*(x1+2*x2)-90 & ((x1+x2)\geqslant 100)\\ 9-(x1+2*x2) & (-90\leqslant(x1+x2)<100)\\ NA & ((x1+x2)\leqslant -90)\end{cases}$$

相应的代码如下：

```
demo.func<-function(x1,x2=0){
    tmp<-x1 + 2*x2
    if( (x1+x2)>=100 ){
        300*tmp-90
    }else if( (x1+x2)>=-90){
        9-tmp
    }else{
       NA
    }
}
```

在上面的代码中，函数名称是 demo.func，输入参数为 x1 和 x2，其中 x2 设置初始数值为 0。在调用该函数时，必须设置 x1，而对于 x2，可以不设置参数，因为此时已被默认设

置了初始值 0。对于函数的返回值，这里并未使用 return() 函数来表示，而是使用了 3 个表达式。注意 tmp<-x1 + 2*x2 虽然也包含了表达式，但是对于赋值语句来说不能作为返回值。上例也可使用如下代码来表达：

```
demo.func<-function(x1,x2=0){
    tmp<-x1 + 2*x2
    if( (x1+x2)>=100 ){
        result<-300*tmp-90
    }else if( (x1+x2)>=-90){
        result<-9-tmp
    }else{
       result<-NA
    }
    result
}
```

在本例中，result 统一作为函数的返回值。对于函数的调用，可以使用以下代码：

```
demo.func(x2=1,x1=1100)
  [1] 330510
demo.func(3,-10000)
  [1] NA
demo.func(1)
  [1] 8
```

上面的代码展示了调用函数的 3 种方式。

- demo.func（x2=1,x1=1100），在设置参数时使用了参数名称 x1 和 x2。这是调用函数最全面的方式。
- demo.func（3,−10000），在设置参数时并未使用参数名称，此时系统自动依据函数定义时参数顺序分配了 x1 和 x2 的设置值。
- demo.func（1），在设置参数时省略了 x2 参数，此时默认把 x2 的初始值设置为 0。

2. 局部变量

在函数内部定义的数据对象（包括以参数形式定义的数据对象）均为局部变量，因此不会改变函数外部同名对象的数值。

```
tmp<-90
demo.func(3,-10000)
  [1] NA
tmp
  [1] 90
```

在上例中，虽然函数内部使用 tmp<− x1 + 2*x2 语句对 tmp 对象进行了赋值，但是对于函数外部定义的同名对象 tmp 不会产生影响。

2.12.3 过程

过程和函数很类似，只是函数内部的对象是局部变量，而过程内部对象的赋值均是全局

性操作。另外，在表现形式上，过程和函数有很大的区别：由于过程内部是全局变量操作，所以输入参数和返回值对于过程而言没有任何意义，即过程既无输入参数的概念，也无返回值的概念，只有函数才有。

使用 expression 函数定义一个过程，这类似于采用自定义函数 function，同时使用 eval 函数来调用该过程。例如：

```
a<-20;b=3
ex <- expression(
if(b>=10){
a<-10+ a
      } else {
          a<-a^2
})
eval(ex)
a   #返回 400
```

上例中定义了一个名为 ex 的过程，过程的内容为 a<-10+ a。然后通过 eval（ex）进行调用，之后对象 a 的数值改变为 400。

2.13 数据的读写等操作

2.13.1 读取数据

在实际处理数据时，往往要对大规模的数据对象进行处理，通过 R 语言可从外部文件中读取原始数据，它所支持的数据格式有很多种，如 Excel 格式、纯文本（TXT）格式、SPSS 格式、SAS 格式等，同时还支持 MySQL 等常用数据库的数据读取。

1. read.table 函数

read.table 函数可以读取外部数据至一个数据框对象中。不过，该函数对于其外部文件的格式往往会有一些特殊的要求。

- 第一行数据要可以作为该数据框各个列的名称。
- 随后的其他条目是各个列的值，并且每行采用相同的分隔符进行分隔。

read.table 函数的基本形式为：

```
read.table(file, header = FALSE, sep = "", encoding = "unknown",…)
```

表 2-30 中给出了具体参数的含义。

表 2-30 read.table 函数参数说明

参数	说明
file	file 是一个字符串，通过它，read.table 函数可以找到欲读取的数据文件
header	如果为 FALSE（默认取值），则不把数据文件的第一行作为列名称处理； 如果为 TRUE，则把数据文件的第一行作为列名称处理
sep	设置用于区分不同列的分隔符，默认为空格。对用制表符分隔的文件使用 sep="\t"

（续）

参数	说明
encoding	编码，以防止出现中文的乱码错误，可以设置 encoding="UTF-8"
as.is	用于设置是否把字符类型的列数据（未转化为实数复数和逻辑类型）转化为因子形式。默认是 FALSE, 即全部转化为因子。如果设置为 T，则所有字符型列数据均不转化为因子。另外，还可以设置不需要转化的列序号
row.names	保存行名的向量 , 或文件中一个变量的序号或名称，缺失时行号取为 1,2,3……
col.names	指定列名的字符型向量（缺失值是 V1, V2, V3, …… ）
na.strings	代表缺失数据的值（转化为 NA）
nrows	可以读取的最大行数（忽略负值）
skip	略过不读取前 n 行数据
strip.white	在 sep 已指定的情况下，如果为 TRUE，则删除字符型变量前后多余的空格
blank.lines.skip	如果为 TRUE，则忽略空白行

例如，以下是一段网站日志文件。

```
数据文件 read.table.data.txt
-----------------------------------------------------------------------
ip time file http_code size
141.243.1.172 [29:23:53:25] "GET /Software.html HTTP/1.0" 200 1497
query2.lycos.cs.cmu.edu [29:23:53:36] "GET /Consumer.html HTTP/1.0" 200 1325
tanuki.twics.com [29:23:53:53] "GET /News.html HTTP/1.0" 200 1014
wpbfl2-45.gate.net [29:23:54:15] "GET / HTTP/1.0" 200 4889
wpbfl2-45.gate.net [29:23:54:16] "GET /icons/circle_logo_small.gif HTTP/1.0"
200 2624
                        --------< 中间数据省略展示 >--------
wpbfl2-45.gate.net [29:23:54:19] "GET /icons/book.gif HTTP/1.0" 200 156
wpbfl2-45.gate.net [29:23:54:19] "GET /logos/us-flag.gif HTTP/1.0" 200 2788
tanuki.twics.com [29:23:54:19] "GET /docs/OSWRCRA/general/hotline HTTP/1.0" 302 0
```

观察上面的数据可以发现，文件内包括了若干行，它对应读取后数据框的行数据。同时可发现该文件中每行均有 4 个空格，这就是列的分隔符。可通过 read.table 函数读取上面的文件，代码如下：

```
root="C:/Users/liming/Desktop/ 写书 /chap2/ 读取 /"
file=paste(root,"read.table.data.txt",sep="")
read.data<-read.table(file,header=TRUE,as.is=c(2,5))
```

其中，root 设置了文件的工作目录，这样在读取多个数据文件时方便修改路径。在 read.table 函数中，header 被设置为 TRUE，依据空格分隔符从第一行中提取出每列的名称依次是 "ip" "time" "file" "http_code" "size"。as.is 设置为向量 c（2,5），表示要求在读取时不要把第 2 和第 5 列转化为因子。

2. read.fwf 函数

当文件内的数据是固定宽度时，可以使用函数 read.fwf 来读取，其语法格式如下：

```
read.fwf(file, widths,sep="\t", as.is = FALSE, …)
```

read.fwf函数的参数与read.table函数很类似，只是read.fwf函数还有一个特殊的widths参数用于设定不同列的宽度。

示例文件如下：

```
数据文件 read.fwf.txt
-------------------------------------------------------------------------------
001 2013.02.12 21:01:37 2013.02.12 21:02:49 005
002 2013.02.12 21:01:39 2013.02.12 21:01:57 001
003 2013.02.12 21:01:41 2013.02.12 21:08:29 011
004 2013.02.12 21:01:41 2013.02.12 21:01:45 001
```

上面文件中的前3位数是序列号，紧接着是一次会话的开始时间和结束时间，最后是本次对话的访问页面数。

下面使用widths参数依次将其分割列宽度为4，20，20，3的数据。对于分割后的数据，系统会自动删除最后的空格。

```
file=paste(root,"read.fwf.txt",sep="")
read.data<- read.fwf(file, widths=c(4, 20, 20,3))
```

3. scan函数

使用scan函数读取外部文件时有如下特点：

- scan函数可以指定输出变量的数据类型。
- 输出对象的类型更灵活，可以是数据框、向量、矩阵、列表。
- 对于很大的数据文件，使用scan函数时读取速度会更快。因为它可以事先设定数据类型，而不是在读取完毕后再检查数据类型的一致性。scan函数的具体参数说明如表2-31所示。

表2-31 scan函数参数说明

参数	说明
file	file是一个字符串，通过它，scan函数可以找到欲读取的数据文件
what	在读取后，用于说明各列数据的类型。可使用logical、integer、numeric、complex、character、raw和list等函数
sep	分隔符，默认为空格
skip	略过不读取前n行数据
nlines	要读取的行数
encoding	编码，为防止中文的乱码错误，可以设置encoding="UTF-8"
na.strings	代表缺失数据的值（转化为NA）

例如，以下是某人的微博发言信息：

```
数据文件 Microtoby-2013-04-22-08-52text.txt
-------------------------------------------------------------------------------
```

```
@benbo007: 当年的家产，如今的国产，唉！国贼难分啊 @yhcqw01: 这海上盛宴，炒来炒去除了能找到一堆文字之外，照片就那么两张，视频没有 / 抠鼻 然后又说板蓝根可以防 H7N9 / 抠鼻 脑残才是解药啊 / 呲牙 / 流汗 @longwu: 飞行棋? @QQmouse: 神马游戏? 女人推动了经大 @yhcqw01: 这海上是个好
--------< 中间数据省略展示 >--------
清明节是谁的节日?【 Win 8 已经失败的 5 个理由 】<a href="http://url.cn/BYYn4m" target="_blank">http://url.cn/BYYn4m</a>
```

观察上面的数据，文件内并未进行分行，同时每段微博开头均有一个 @ 字符串用于分割。下面利用 scan 函数来读取相应的数据，代码如下：

```
file=paste(root,"Microtoby-2013-04-22-08-52text.txt",sep="")
weibo<-scan(file,what="character",sep="@",encoding="UTF-8" )
```

这里使用 @ 字符串分隔 file 内的数据，然后返回一个 character 字符类型的向量。

4. read.csv 函数

csv（逗号分隔）文件是较为常用的数据文件格式。图 2-21 是订单取消的原因数据，文件名为 " 订单取消原因分析（3 1-5 31）.csv"。

	A	B	C	D	E	F	G	H	I	J	K
1	测试订单	等待太久	订单重复	订购错误	更换其他商	价格太贵	忘选优惠品	已在别处购	有缺货商品	支付失败	其他
2	311	43	251	571	336	29	154	22	57	298	184
3	341	22	210	541	263	13	144	22	49	224	137
4	234	42	215	527	263	27	127	24	50	220	116
5	193	44	153	494	261	14	153	30	38	215	137
6	109	29	155	443	208	19	120	26	36	248	98
7	100	39	189	487	191	20	95	23	36	276	94
8	228	29	170	545	250	13	190	21	52	231	142
9	173	33	192	536	228	16	146	25	60	230	142
10	293	30	184	525	292	19	152	32	36	208	152
11	259	29	172	525	255	21	176	17	50	238	162
12	241	47	226	526	276	23	157	27	34	305	163
13	120	27	260	575	256	26	159	28	222	234	141
14	130	20	462	631	296	18	158	30	50	356	164
15	256	125	248	714	373	27	183	23	59	300	166
16	346	212	287	728	363	42	239	33	64	280	209
17	291	198	1055	1227	493	25	333	28	195	328	268

图 2-21 数据集截图

通过 read.csv 函数可以读取 csv 文件并返回一个数据框对象。例如：

```
file=paste(root," 订单取消原因分析 (3 1-5 31).csv",sep="")
data.csv<-read.csv(file)
```

返回的 data.csv 一是个数据框对象。

5. Excel 格式文件的读取

使用 RODBC 包可以读取 Excel 格式文件，其核心函数分别是 odbcConnectExcel 和 sqlFetch。

- odbcConnectExcel（dsn）表示打开 Excel 文件功能，其中 dsn 用于指定 Excel 文件路径。打开后会返回一个文件话柄（channel），后续函数将使用该话柄进行其他操作。当全部操作完成后，需要使用 close 函数关闭该文件。
- sqlFetch（channel, sqtable）可以指定读取 Excel 文件的某个 sheet 内的数据内容。参数 channel 是 odbcConnectExcel 函数的返回值，参数 sqtable 是 Excel 文件内的 sheet 名称。

相应的示例代码如下：

```
library (RODBC)
excel_file <- odbcConnectExcel("D:/r/lab/data/grades.xls")
sheet_grades <- sqlFetch ( excel_file,"grades")
close ( excel_file )
```

odbcConnectExcel 函数读取原数据文件 grades.xls 并返回一个文件话柄 excel_file，然后使用 sqlFetch 函数读取 grades.xls 文件（文件话柄 excel_file 作为其输入参数）中名为 grades 的 sheet 内数据，返回值 sheet_grades 是一个数据框对象。当所有数据均读取完后，使用 close 函数关闭文件话柄 excel_file。

2.13.2 输出数据

1. write 函数

对于向量和矩阵数据的输出，一般使用 write 函数写入某一文件中。其基本形式为：

```
write (x,file, append=FALSE)
```

表 2-32 中给出了具体参数的含义。

表 2-32 write 函数参数说明

参数	说明
x	数据源，一般是向量或矩阵类型的数据对象
file	输出的文件
append	如果是 FALSE（默认的取值），则清空 file 文件内的原有数据，再写入 x 的内容； 如果是 TRUE，则在 file 文件原有内容的基础上追加写入 x 的内容

2. write.table 以及 write.csv 函数

对于列表或数据框类型对象，建议使用 write.table 写入文本文件中，或者使用 write.csv 写入 csv 文件中。其基本参数与 write 类似，也涉及了 x/file/append。

3. 输出 R 命令至外部文件

cat 函数可以把 R 语言命令输出至一个外部文件，然后可通过 source 函数运行该批处理文件。基本形式为：

```
cat (…, file = "", sep = " ", append = FALSE)
```

其中，…是预输出的命令，其他参数可以参考 write 函数的说明。

第 3 章

R 语言的绘图基础

3.1 概述

R 语言除了拥有良好的数据处理和分析能力外，对于数据的展现也有极其灵活和强大的应用。由于图形对于分析结果的表达往往更具有直观性和简单性，所以对于一份优秀的数据分析报告而言，把数据结果以适当的图形方式展示后，其沟通效果和说服力会更佳。

那么，R 语言的绘图函数又有哪些呢？来看看表 3-1。

表 3-1 R 语言绘图函数分类

	分类	描述
1	绘图窗口操作函数	新建绘图窗口、设置当前活动窗口等操作
2	高级绘图函数	如果 add=FALSE（默认），则在新窗口中创建一个图形； 如果 add=TRUE，则在当前活动窗口中图形叠加在原有的图形之上
3	低级绘图函数	在现有活动窗口中添加点、线、文字等图形绘图元素，以及图例、坐标轴标题、正标题、副标题等的提示说明绘图元素
4	扩展绘图包函数	RColorBrewer、lattice、ggplot2 扩展包的核心绘图函数

为了更好地理解这些函数，下面分析某网站的 PV 趋势图，并剖析其中包含的绘图元素，如图 3-1 所示。

在图 3-1 中，各个数字标识所代表的绘图元素如表 3-2 所示。

3.2 颜色以及文字 / 点 / 线参数的设置

上面简单介绍了添加绘图元素的常用方法。添加图形元素后，还要对其属性进行设置。表 3-3 是常用的属性，一般相关的函数均有这些属性。

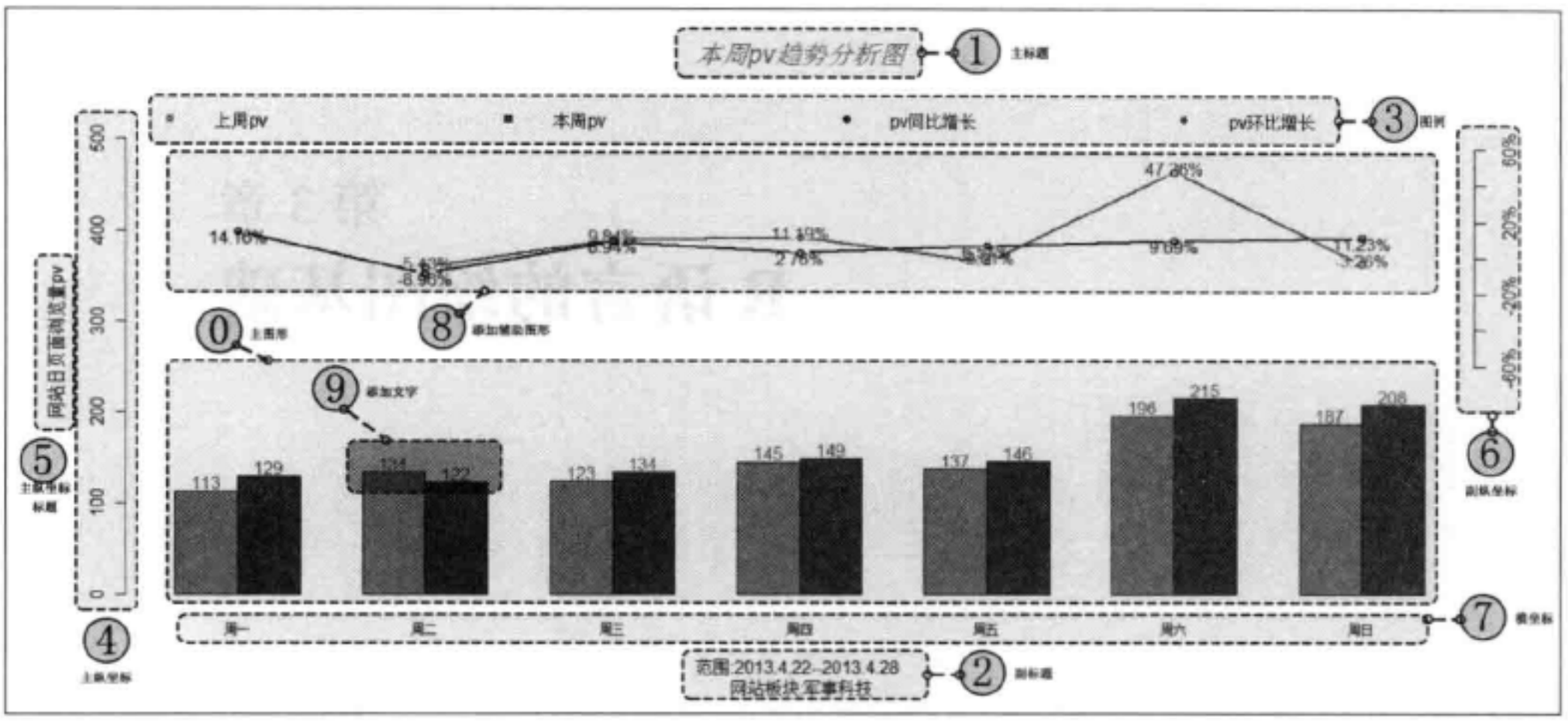


图 3-1 网站 PV 趋势图

表 3-2 引例中各种绘图元素的说明

对应位置	绘图元素		设置方式	
			高级绘图函数及其参数	低级绘图函数及其参数
0	主图形		高级绘图函数且 add=FALSE（默认）	无
1	正标题		main 参数	title 函数的 main 参数
2	副标题		sub 参数	title 函数的 sub 参数
3	图例		不同函数对应的参数不同	legend 函数
4	主纵坐标		ylim, axes 等参数	axis 函数
5	主纵坐标标题		ylab 参数	title 函数的 ylab 参数
6	副纵坐标		无	axis 函数
	副纵坐标标题		无	mtext 函数
7	横坐标		xlim, axes 等参数	axis 函数
	横坐标标题		xlab 参数	title 函数的 xlab 参数
8	辅助图形	图形	高级绘图函数且 add=TRUE	无
		点	无	points 函数
		线	可绘制线的高级绘图函数且 add=TRUE	lines（添加曲线）函数 abline（添加直线）函数
9	文字		无	text 函数

表 3-3 文字、点、线的常用属性

设置对象	属性	常用英文简写
文字	颜色	col
	文本字体样式	font
	大小（缩放倍数）	cex

（续）

设置对象	属性	常用英文简写
点	点的符号	pch
	颜色	col
	大小（缩放倍数）	cex
线	线条宽度	lwd
	线条样式	lty
	颜色	col

3.2.1 颜色

通过设置 col 参数，可以改变图像、坐标轴、文字、点、线等的颜色参数。

1. 固有颜色

R 语言提供了自带的固定种类的颜色，主要涉及的函数是 colors()。该函数可以生成 657 种颜色名称，代表 657 种颜色，具体如下：

```
colors()
  [1] "white"                "aliceblue"            "antiquewhite"
  [4] "antiquewhite1"        "antiquewhite2"        "antiquewhite3"
  [7] "antiquewhite4"        "aquamarine"           "aquamarine1"
                ·····< 此处由于数据过多，省略显示 >·····
[649] "wheat3"               "wheat4"               "whitesmoke"
[652] "yellow"               "yellow1"              "yellow2"
[655] "yellow3"              "yellow4"              "yellowgreen"
```

在设置 col 参数时，直接填写相关颜色的代表文字便可。例如：

```
plot(c(1:10),type="b",col="black")      # 设置线的颜色
text(6,5,labels=" 文字 ",col="red")      # 设置文字的颜色
points(2,5,col="blue")                  # 设置点的颜色
```

不过，单纯地获知颜色名称而不知道其所对应的颜色样式是没有意义的，所以还要让两者对应起来。可以运行下面的代码，展示所有 657 种颜色名称及其对应的颜色，并输出在一个 PDF 文件中。

```
pdf("D:/colors-bar.pdf", height = 120)
par(mar = c(0, 10, 3, 0) + 0.1, yaxs = "i")
barplot(rep(1, length(colors())), col = rev(colors()),
+    names.arg = rev(colors()), horiz = TRUE, las = 1,
+    xaxt = "n", main = expression("Bars of colors in" ~
+    italic(colors())))
dev.off()
windows
      2
```

运行后，在计算机的 D 盘下会出现一个名为 colors-bar.pdf 的文件，其部分展示如图 3-2 所示。

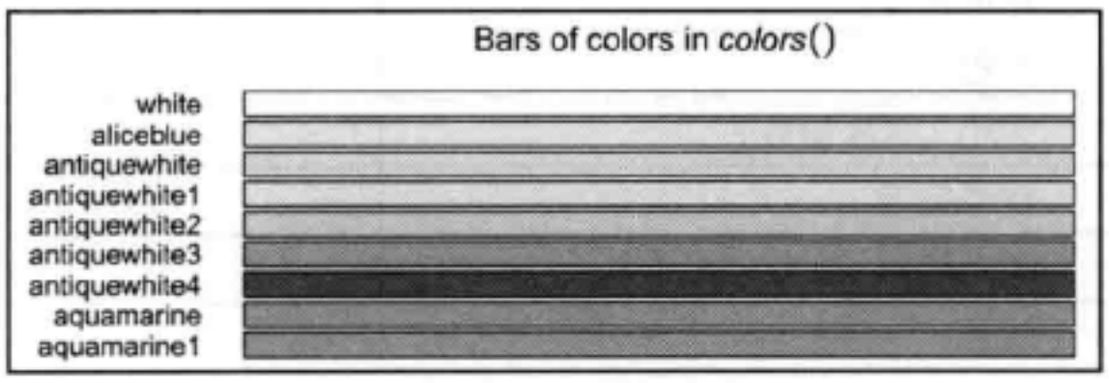


图 3-2 colors-bar.pdf 部分截图

有了此输出，颜色名称与具体颜色样式也就清楚了，用户可以依据 PDF 文件内的颜色样式找到相应的颜色名称，从而设置相关颜色参数。

2. RGB 取色

颜色的搭配很有讲究，一些初级用户很难直接从 colors() 的 657 种颜色中找到一组合适的颜色搭配。这时可以学习别人的配色方案，并通过取色工具设置相应的颜色。来看一个实例。

1）使用取色软件获取上文包括 657 种颜色样式的 PDF 文件中的 aquamarine1 颜色。

aquamarine1:

获取的 RGB 依次为：红色 =126，绿色 =255，蓝色 =211

RGB 颜色的十六进制数值为 7EFFD3。

2）使用 rgb 函数，把 RGB 颜色转化为十六进制数值。

```
Hex<-rgb(red=126,green=255,blue=211,max = 255);Hex
[1] "#7EFFD3"
```

3）设置 col 参数。

```
text(6,5,labels=" 文字 ",col= Hex)
```

补充，如果取色工具可以直接读取出 RGB 的十六进制数值，则可以直接使用 text（6,5,labels=" 文字 ",col= "#7EFFD3"）语句。

3. 主题调色板

上面两种方式均要求用户选择和寻找较好的配色方案。其实 R 本身也提供特定颜色主题的配色方案。这些配色方案用一系列渐变的颜色表现特定的主题。

（1）rainbow()

rainbow() 即彩虹的颜色，是由“赤、橙、黄、绿、青、蓝、紫”一系列颜色组成的。该函数的用法如下：

```
rainbow(n,s=1,v=1,start=0,end=max(1,n-1)/n,gamma=1);
```

其中涉及的具体参数如表 3-4 所示。

表 3-4 rainbow 函数的参数说明

参数	说明
n	设定产生颜色的数目
start 和 end	设定彩虹颜色的一个子集，生成的颜色将从这个子集中选取，这个子集选取的大致分界线为： 赤色（red）为 start=0，end=1/6 黄色（yellow）为 start=1/6，end=2/6 绿色（green）为 start=2/6，end=3/6 青色（cyan）为 start=3/6，end=4/6 蓝色（blue）为 start=4/6，end=5/6 紫色（magenta）为 start=5/6，end=1

了解了 rainbow() 的基本属性，那如何使用呢？来看一个实例，在该例中会产生由赤、橙、黄、绿、青、蓝、紫 7 色依次合成 6 组颜色，每组包含 6 个渐变色。

```
barplot(rep(1,41),
        col=c(rainbow(n=6,start=0,end=1/6),
              "white",
              rainbow(n=6,start=1/6,end=2/6),
              "white",
              rainbow(n=6,start=2/6,end=3/6),
              "white",
              rainbow(n=6,start=3/6,end=4/6),
              "white",
              rainbow(n=6,start=4/6,end=5/6),
              "white",
              rainbow(n=6,start=5/6,end=1)))
x.text<-c("赤色","黄色","绿色","青色","蓝色","紫色")
axis(side=1,at=seq(from=3,length.out=6,by=8.5),labels =x.text,cex=0.75)
```

结果如图 3-3 所示。

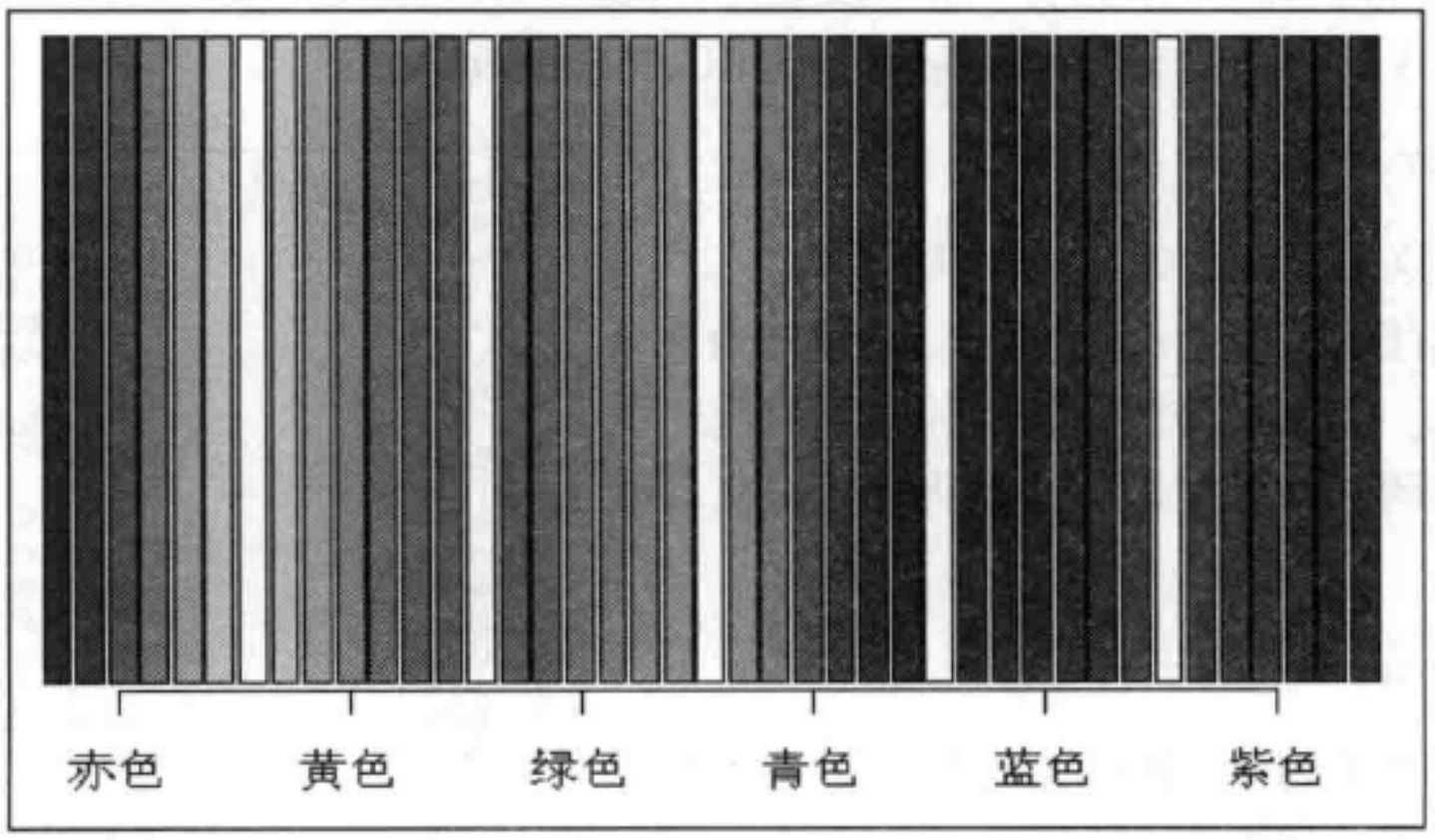
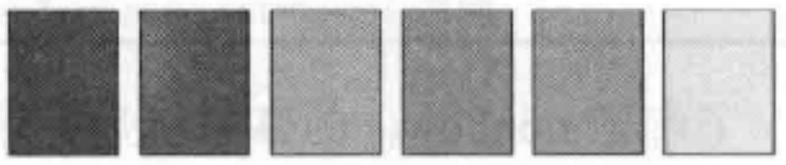


图 3-3 使用 rainbow() 的结果图示

（2）其他主题配色函数

了解了 rainbow() 的用法，再来看看其他的主题配色函数，如表 3-5 所示。

表 3-5 其他配色函数

配色函数	说明	参数	实例
heat.colors()	从红色渐变到黄色，再变到白色	n：设定产生颜色的数目	barplot（rep（1,6）,col=heat.colors（6））
terrain.colors()	从绿色渐变到黄色，再到棕色，最后到白色		barplot（rep（1,6）,col=terrain.colors（6））
topo.colors()	从蓝色渐变到青色，再到黄色，最后到棕色		barplot（rep（1,6）,col=topo.colors（6））
cm.colors()	从青色渐变到白色，再到粉红色		barplot（rep（1,6）,col=cm.colors（6））

4. RColorBrewer 颜色扩展包

RColorBrewer 包提供了 3 套很好的配色方案。用户只需要指定配色方案的名称，就可以用包中的 brewer.pal() 函数生成颜色。这 3 套配色方案包括：

- 连续型 Sequential：生成一系列连续渐变的颜色，通常用来标记连续型数值的大小。
- 极端型 Diverging：生成用深色强调两端、浅色标示中部的系列颜色，可用来标记数据中的离群点。
- 离散型 Qualitative：生成一系列彼此差异比较明显的颜色，通常用来标记分类数据。

在使用这 3 套配色方案之前，先要加载 RColorBrewer 包。

```
library(RColorBrewer)
```

下面介绍这 3 套配色方案的用法。

1）seq 连续型：共 18 组颜色，每组分为 9 个渐变颜色展示。使用渐变色往往能让图形看起来更美观，避免单调的颜色在图形中显得突兀。实现代码如下：

```
display.brewer.all(type = "seq")
```

以上代码展示的 seq 连续型配色方案中各组颜色的名称和样式，如图 3-4 所示。

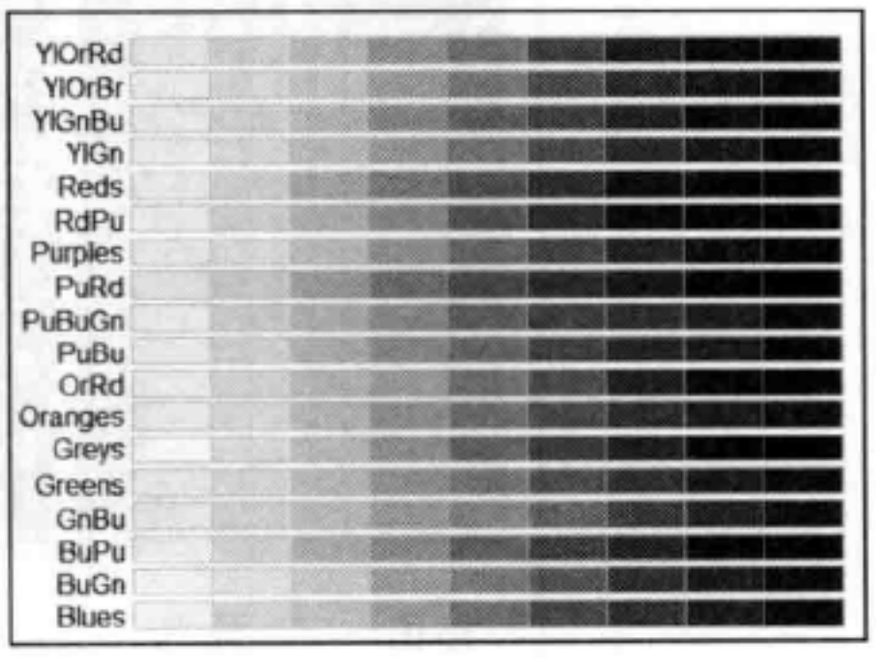


图 3-4 seq 连续型配色方案中各组颜色的名称和样式

如果想使用 YlOrRd 组的第 3 ～ 8 种颜色

()，则可用如下代码来实现。

```
barplot(rep(1,6),col=brewer.pal(9, "YlOrRd")[3:8])
```

该代码的实现效果如图 3-5 所示。

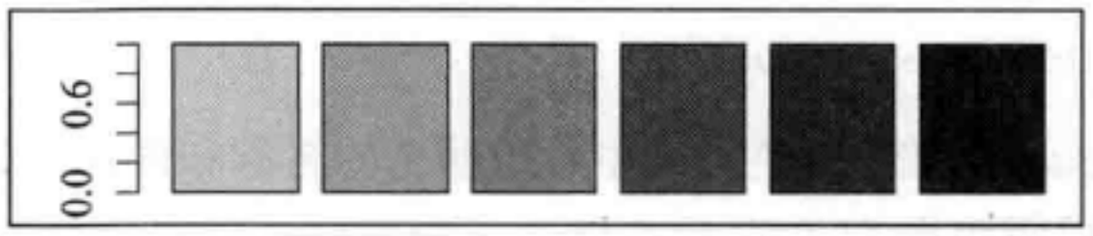


图 3-5 YlOrRd 组的第 3 ～ 8 种颜色

从上例可知，可以使用 brewer.pal（9, "< 某组渐变颜色的名称 >"）来获取该组渐变色的全部 9 种颜色。

2）div 极端型：共 9 组颜色，每组分为 11 个渐变颜色展示。其实现代码如下：

```
display.brewer.all(type = "div")
```

上述代码展示的 div 极端型配色方案中各组颜色的名称和样式，如图 3-6 所示。

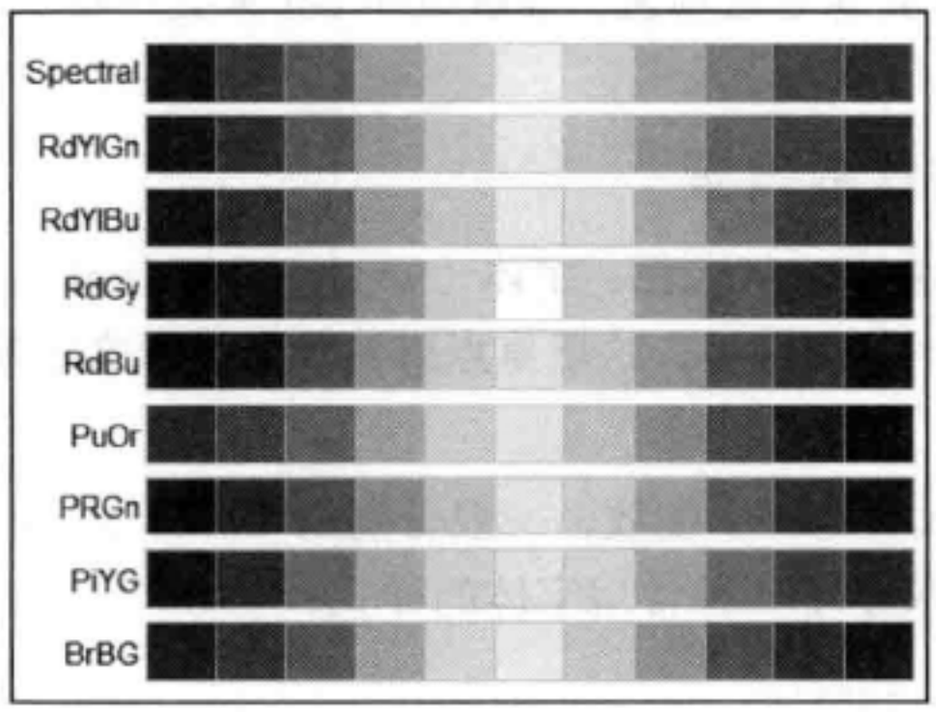


图 3-6 div 极端型配色方案中的各组颜色

如果想使用 BrBG 组的第 3 ～ 8 种颜色（），则可用如下代码来实现。

```
barplot(rep(1,6),col=brewer.pal(11, "BrBG")[3:8])
```

该代码的实现效果如图 3-7 所示。

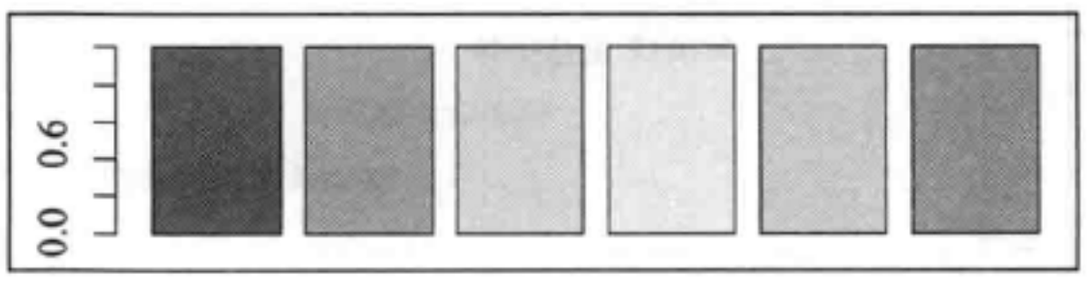


图 3-7 BrBG 组的第 3 ～ 8 种颜色

和 seq 连续型渐变色的使用不同，由于极端型中每组颜色分为 11 个渐变颜色，因此 brewer.pal 函数第一个参数不再是 9，而是 11，即应使用 brewer.pal（11, "<某组渐变颜色的名称>"）来读取该组渐变的 11 颜色。

3）qual 离散型：共 8 组颜色，每组渐变颜色数也不尽相同。其实现代码如下：

```
display.brewer.all(type = "qual")
```

上述语句展示的 qual 离散型配色方案中各组颜色的名称和样式，如图 3-8 所示。

在该类型中，使用每组渐变色的方式与 seq 连续型和 div 极端型类似：通过 brewer.pal（n, "<某组渐变颜色的名称>"）语句可以读取该组内的 n 个渐变色（其中 n 是该组内渐变色的数目）。

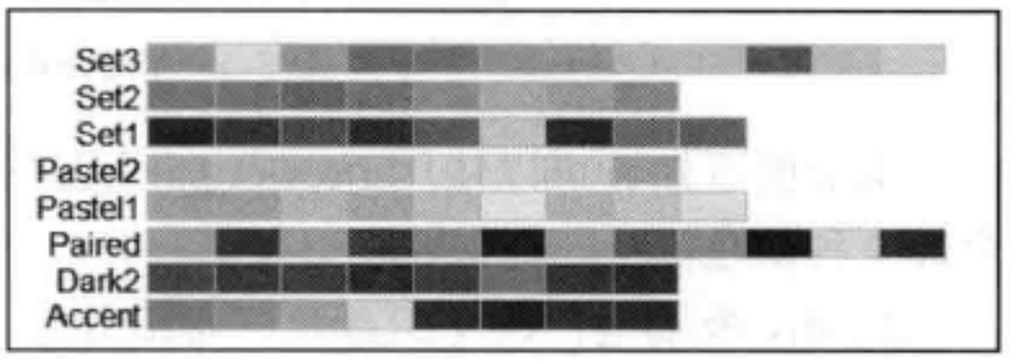


图 3-8 qual 离散型配色方案中各组的颜色

注意

一般的绘图函数会使用 col 颜色参数。此外，一些元素还可以使用 bg 参数设置其背景颜色，使用 border 参数设置其边框颜色，其赋值和 col 参数一样。

3.2.2 文字元素相关参数设置

文字元素可以设置的参数一般包括：字体（font）、颜色（col）、大小（缩放倍数 cex）等，其中的颜色设置已在 3.3.1 节中讲解，这里重点讲解如何设置文字的字体和大小。

1. 字体

font（字体）参数的取值是一个整数，一般分别用 1、2、3、4 来表示正常体、粗体、斜体和粗斜体 1。以下代码展示了 4 种常用字体的样式：

```
plot(c(0:5),col="white")
text(2,4,labels="font=1:正常字体（默认）",font=1)
text(3,3,labels="font=2:粗体字体",font=2)
text(4,2,labels="font=3:斜体字体",font=3)
text(5,1,labels="font=4:粗斜体字体",font=4)
```

结果展示如图 3-9 所示。

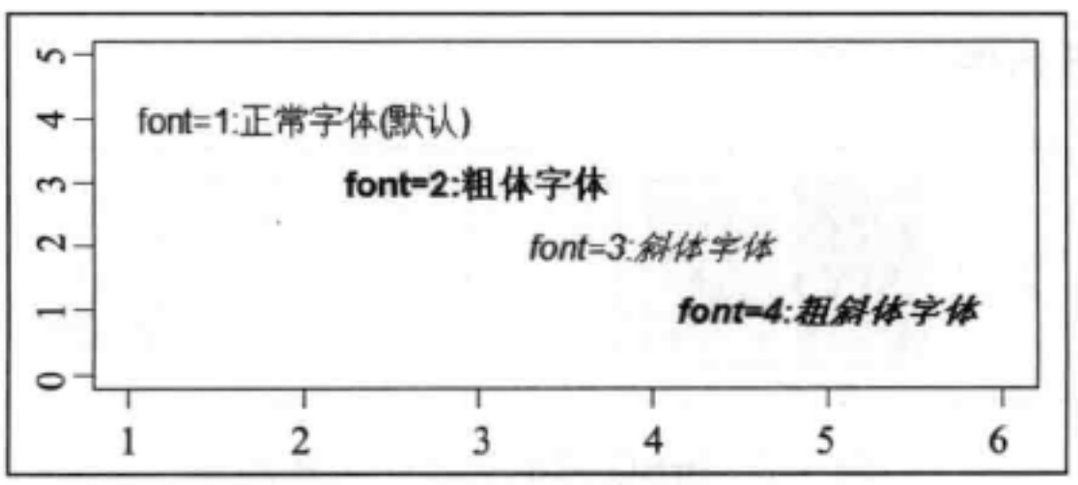


图 3-9 设置不同 font 参数值时的样式

2. 大小

cex（缩放倍数）参数的取值是一个实数，默认为 1，表示不缩放。取值小于 1 时，表示缩小。取值大于 1 时，表示放大。以下代码展示了 cex 分别取 0.5、0.8、1、1.2、1.5 时的大小样式。

```
plot(c(0:6),col="white",xlim=c(1,8))
text(2,5,labels="cex=0.5: 放大 0.5 倍 ",cex=0.5)
text(3,4,labels="cex=0.8: 放大 0.8 倍 ",cex=0.8)
text(4,3,labels="cex=1( 默认 ): 正常大小 ",cex=1)
text(5,2,labels="cex=1.2: 放大 1.2 倍 ",cex=1.2)
text(6,1,labels="cex=1.5: 放大 1.5 倍 ",cex=1.5)
```

结果展示如图 3-10 所示。



图 3-10　设置不同 cex 参数值时的样式

3.2.3　点元素相关参数设置

点元素可以设置的参数一般包括：点样式（pch）、颜色（col）、大小（缩放倍数 cex）等，其中的颜色设置已在 3.3.1 中讲解，这里只做简要说明。

1. 点样式和颜色

pch（点样式）参数可取 1 ～ 25 的数字以及部分符号，下面的代码展示了 pch 取值为 1 ～ 25 时的样式。

```
plot(1,col="white",xlim=c(1,9),ylim=c(1,7))
for(i in c(0:25)){
    x<-(i%/%5)*1+1
    y<-6-(i%%5)
    if(length(which(c(21:25)==i)>=1)){#21--25 的点格式可以设置背景颜色。
        points(x,y,pch=i,bg="red",cex=2)
    }else{
        points(x,y,pch=i,cex=2)
    }
    text(x+0.2,y+0.2,labels=paste("pch=",i))
}
```

结果显示如图 3-11 所示。

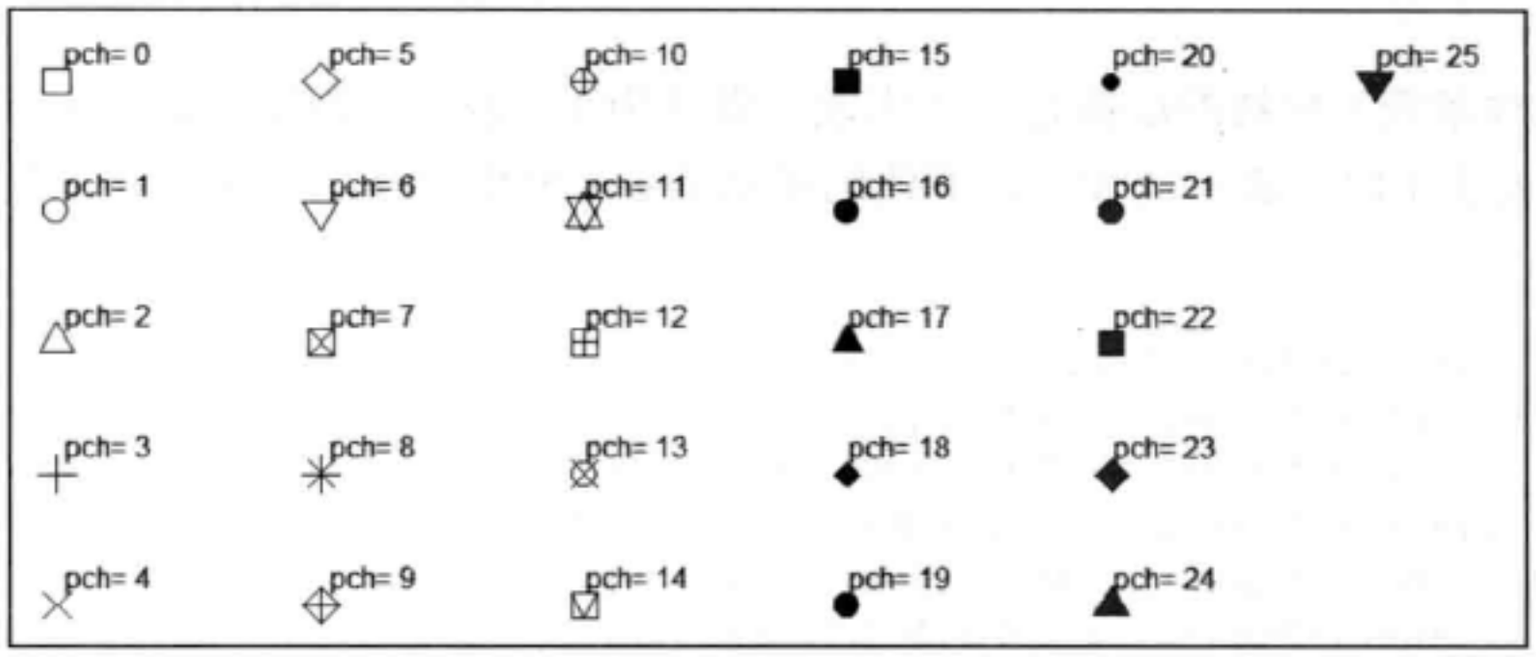


图 3-11　设置不同 pch 参数值时的样式

注意，同样是正方形 / 圆 / 上三角 / 倒三角 / 菱形，pch 取 0 ～ 14 时，其点是空心点，可以用 col 参数设置其边框的颜色；pch 取 15 ～ 20 时，其点是实心点，可以用 col（颜色）参数设置其填充的颜色；pch 取 21 ～ 25 时，其点也是实心点，但是可以分别用 col 参数设置其边框的颜色，并且可用 bg 参数设置其内部填充的颜色。

此外 pch 的取值还可以为：“* . o O 0 + - |”，具体如下：

```
points(6,5,pch="*",cex=2);text(6.2,5.2,labels=paste("pch=\"*\""))
points(6,4,pch=".",cex=2);text(6.2,4.2,labels=paste("pch=\".\""))
points(6,3,pch="o",cex=2);text(6.2,3.2,labels=paste("pch=\"o\""))
points(6,2,pch="O",cex=2);text(6.2,2.2,labels=paste("pch=\"O\""))
points(7,6,pch="0",cex=2);text(7.2,6.2,labels=paste("pch=\"0\""))
points(7,5,pch="+",cex=2);text(7.2,5.2,labels=paste("pch=\"+\""))
points(7,4,pch="-",cex=2);text(7.2,4.2,labels=paste("pch=\"-\""))
points(7,3,pch="|",cex=2);text(7.2,3.2,labels=paste("pch=\"|\""))
```

效果如图 3-12 所示。

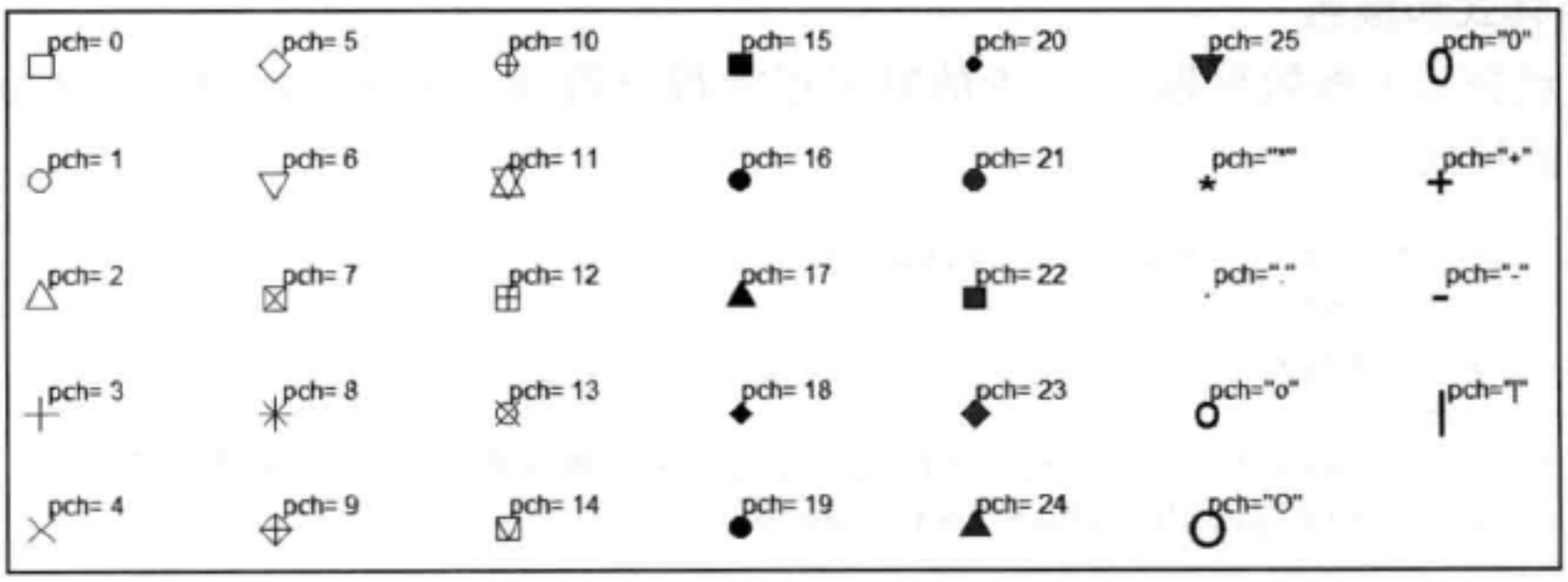


图 3-12　设置不同 pch 参数值时的样式

2. 大小

大小（cex）参数的设置与文字元素的大小参数设置一致，这里就不再详述了。

3.2.4 线元素相关参数设置

线元素可以设置的参数一般包括：线条样式（lty）、颜色（col）、粗细（lwd）等。

1. 线条样式

线条样式（lty）主要是指线实线、虚线、点线、点划线等的样式。lty 参数设置为不同数值和字符串对应的线条样式如表 3-6 所示。

表 3-6 lty 设置为 0 ～ 6 和字符串对应的线条样式

数值	字符串	说明	数值	字符串	说明
0	"blank"	不画线	4	"dot-dash"	点画线
1	"solid"	实线	5	"longdash"	长画线
2	"dashed"	虚线	6	"twodash"	点长画线
3	"dotted"	点线			

运行下面的代码，可以查看 lty 参数取值为 0 ～ 6 时的样式。

```
data<-matrix(rep(rep(1:7),10),ncol=10,nrow=7)
plot(data[1,],type="l",lty=0,ylim=c(1,8),xlim=c(-1,10),axes=F)
text(0,1,labels="lty=0")
for(i in c(2:7)){
    lines(data[i,],lty=i-1)
    text(0,i,labels=paste("lty=",i))
}
```

结果如图 3-13 所示。

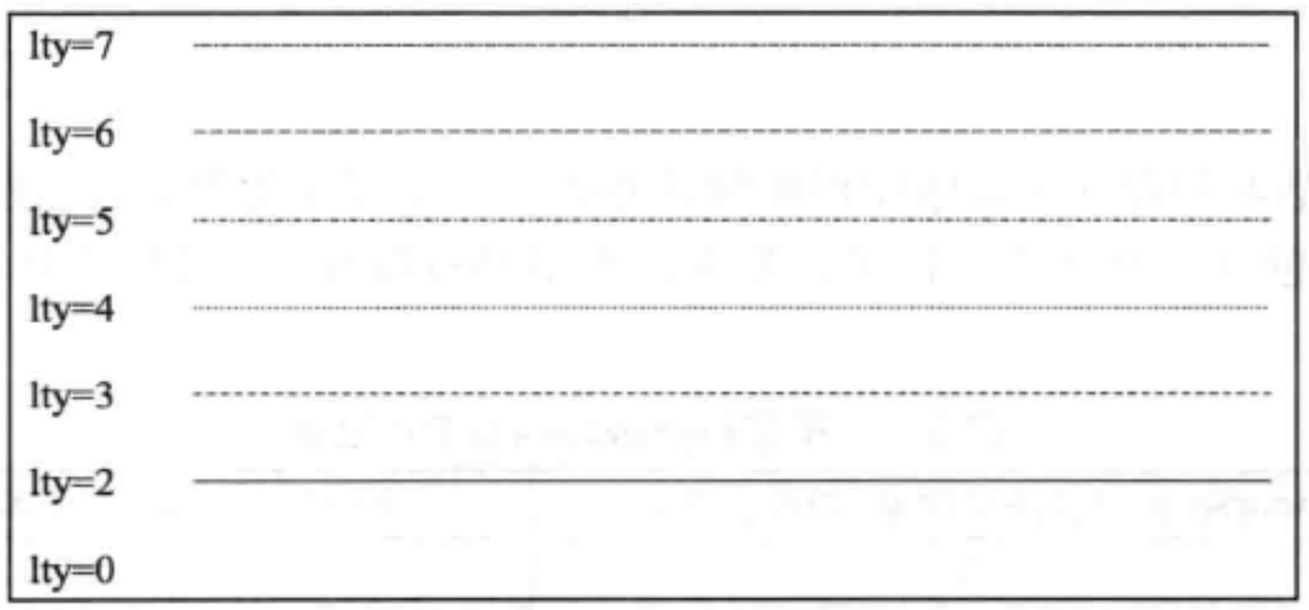


图 3-13 设置不同 lty 参数值时的样式

2. 线条宽度

文字和点有大小的概念，而线条则有宽度的概念。lwd（宽度）参数默认为 1，表示不缩放。取小于 1 时，表示缩小。取大于 1 时，表示放大。以下代码展示了 lwd 分别取 0.5、0.8、1、1.5、2 时的线条宽度：

```
data<-matrix(rep(rep(1:5),10),ncol=10,nrow=5)
plot(data[1,],type="l",lwd=0.5,ylim=c(1,8),xlim=c(-1,10),axes=F);
```

```
text(0,1,labels="lwd=0.5")
lines(data[2,],lwd=0.8);text(0,2,labels="lwd=0.8")
lines(data[3,],lwd=1);text(0,3,labels="lwd=1")
lines(data[4,],lwd=2);text(0,4,labels="lwd=2")
lines(data[5,],lwd=4);text(0,5,labels="lwd=4")
```

结果如图 3-14 所示。

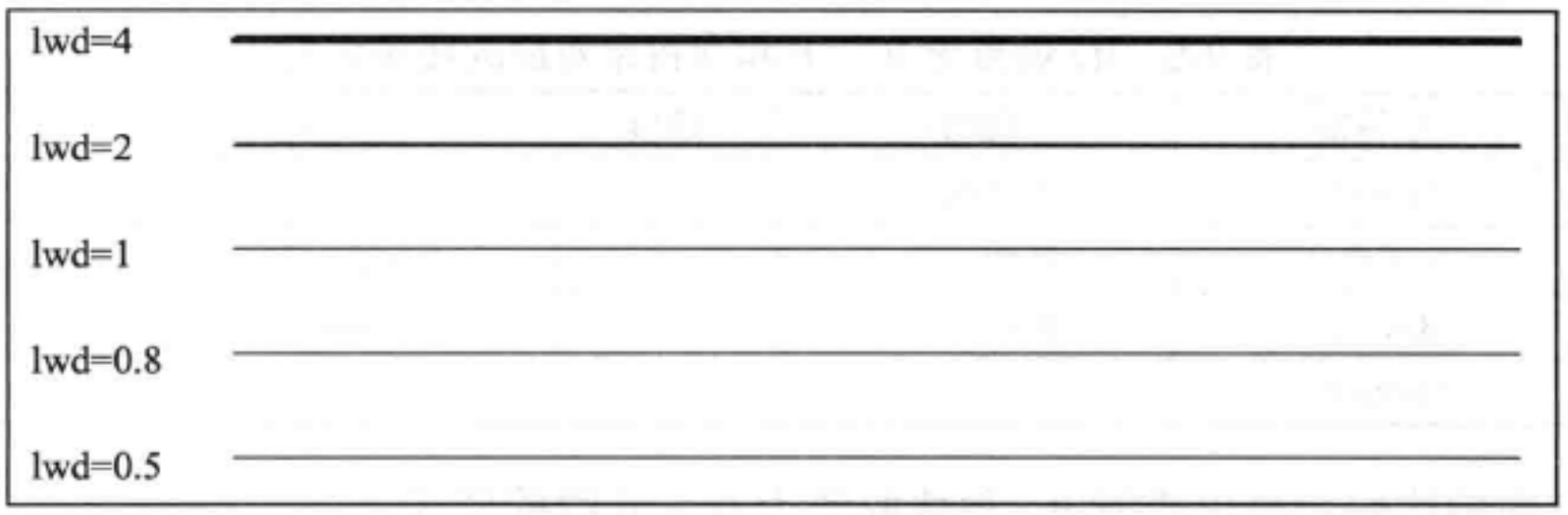


图 3-14 设置不同 lwd 参数值时的样式

由图 3-14 可知，线条缩小的效果不太明显。

3.3 低级绘图函数

低级绘图函数主要是在现有活动窗口中添加点、线、文字等图形绘图元素，以及图例、坐标轴标题、正标题、副标题等提示说明绘图元素。

3.3.1 引例

因为低级函数必须存在于高级绘图函数的基础上，所以在使用低级绘图函数前，要先使用高级绘图函数创建一个图形。这里以某网站军事科技版块的日 PV 趋势图为例加以说明，如表 3-7 所示。

表 3-7 军事科技版块的日 PV 数据

date	data$pre（上周 PV 数据，单位：万）	data$now（本周 PV 数据，单位：万）
周一	113	129
周二	134	122
周三	123	134
周四	145	149
周五	137	146
周六	196	215
周日	187	208

下面利用以上数据制作一个条形图，并依据上周和本周数据分两组进行展示。代码如下：

```
data<-data.frame(pre=c(113,134,123,145,137,196,187),
                 now=c(129,122,134,149,146,215,208));
ylim.max<-550
col=c("azure4","brown4")
barplot(as.matrix(rbind(data$pre,data$now)),
        beside=TRUE,ylim=c(0,ylim.max),col=col,axes=F)
axis(2)
```

产生的图形如图 3-15 所示。

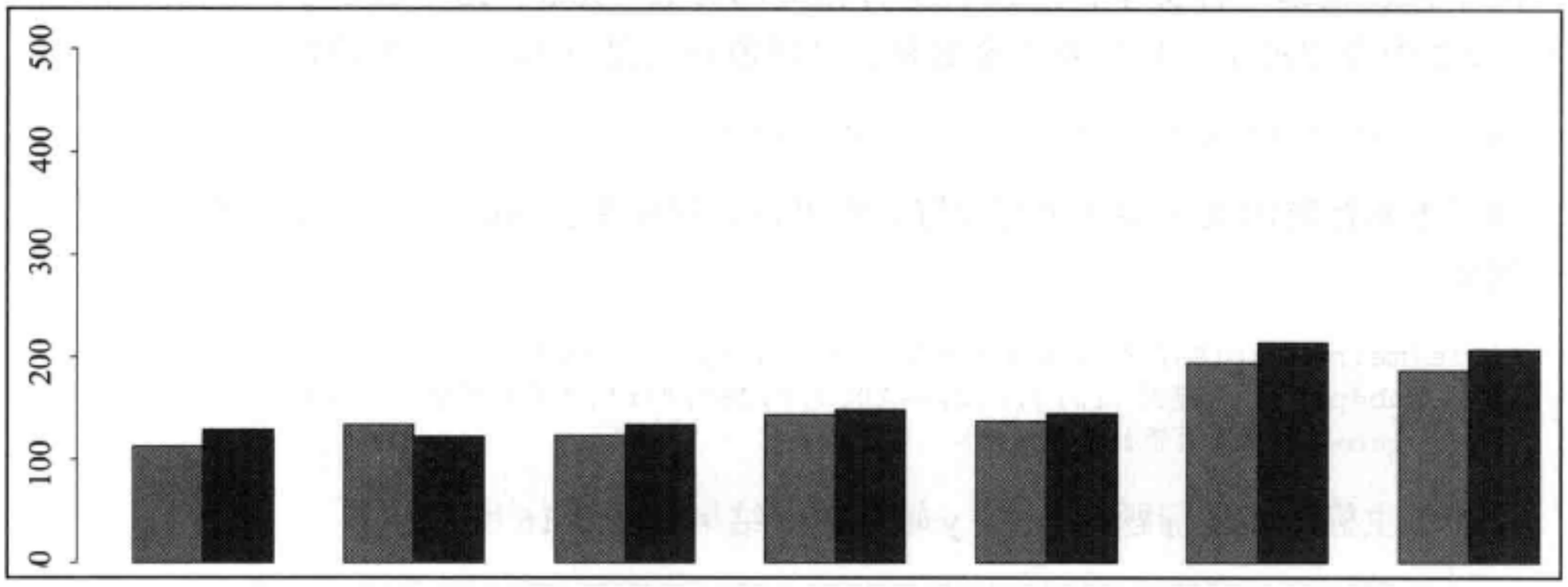


图 3-15　网站 PV 引例图 1

在上面的代码中，data 是数据框对象，其 pre 列是上周 7 天的日 PV 数据，now 列是本周 7 天的日 PV 数据。barplot 是绘制条形图的高级绘图函数，其样本数据要以矩阵形式输入 as.matrix（rbind（data$pre,data$now）），转化后数据为：

```
     [,1]  [,2]  [,3]  [,4]   [,5]  [,6]  [,7]
[1,]  113   134   123   145   137   196   187
[2,]  129   122   134   149   146   215   208
```

本次绘图主要是处理上周（第一行）和本周（第二行）的共两组数据。beside 参数设置为 TRUE，表示由两组数据得到的柱状图是水平并列排列的。如果设置为 FALSE，则条形图是垂直排列的。barplot 中涉及的其他参数，会在 3.4.5 节中详细讲解。

3.3.2　标题

标题元素主要包括主标题、副标题、x 轴标题、y 轴标题。下面介绍其中涉及的函数及其用法。

1. title 函数

title 常用参数如表 3-8 所示。

表 3-8 title 函数的常用参数

参数	说明	实例
main	设置主标题内容和文字属性	main=" 主标题 " main=list（" 主标题 ",font=3,col="red",cex=1.5）
sub	设置副标题内容和文字属性	sub=list（" 复标题 ",font=3,col="red",cex=1.2）
xlab	设置 x 轴标题内容和文字属性	xlab=list（"x 轴标题 ",font=3,col="red",cex=0.75）
ylab	设置 y 轴标题内容和文字属性	ylab=list（"y 轴标题 ",font=3,col="red",cex=0.75）

由于标题也是一种文字，所以它也有相应的参数（font、col、cex 等）设置，相关内容已在 1.2.2 中介绍过了。只是要注意的是，其赋值形式是以 list 形式承载的。例如：

```
main=list(" 主标题 ",font=3,col="red",cex=1.5)
```

如果不对标题的文字参数进行设置，则可以直接使用：main=" 主标题 " 形式。例如：

```
title(main=list(" 本周 pv 趋势分析图 ",cex=1.5,col="red",font=3),
      sub=paste(" 范围 :2013.4.22--2013.4.28","\n"," 网站板块 : 军事科技 "),
      ylab=" 网站日页面浏览量 pv")
```

添加了主标题、副标题和（主）y 轴标题，结果如图 3-16 所示。

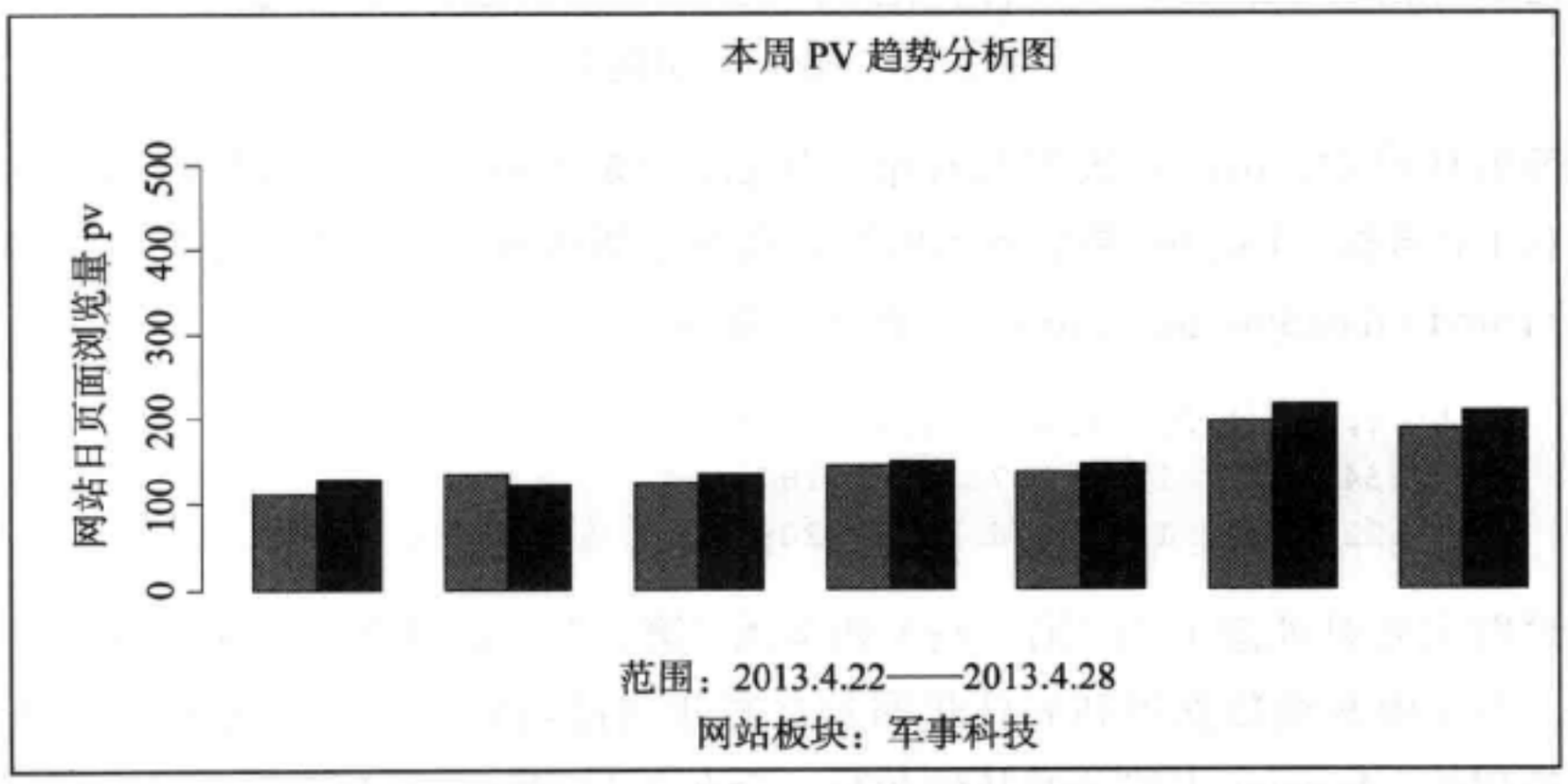


图 3-16 网站 PV 引例图 2

这里需要注意的是，副坐标通过使用回车特殊字符“\n”设置为两行。在 R 中，特殊字符均使用反斜杠“\”来开头。

2. 在绘图函数中直接设置参数

对于上一个要点中给出的案例，也可以直接在高级绘图函数 barplot 中设置相关参数。代码如下：

```
barplot(as.matrix(rbind(data$pre,data$now)),
```

```
beside=TRUE,ylim=c(0,ylim.max),col=col,axes=F,
        main=list("本周pv趋势分析图",cex=1.5,col="red",font=3),
        sub=paste("范围:2013.4.22--2013.4.28","\n","网站板块:军事科技"),
        ylab="网站日页面浏览量pv")
```

3.3.3 图例

当图表中承载了多组数据用于比较时，通常会使用不同的颜色来区分，并且使用图例说明不同颜色代表的数据，如图 3-17 所示。

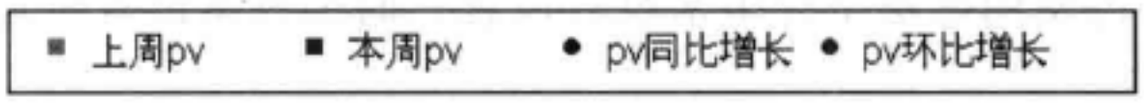


图 3-17 图例

图例中既包括文字，也包括点和线元素。R 语言中的图例可以使用 legend 函数和高级绘图函数的图例参数进行设置。

1. legend 函数

legend 函数的常用参数如表 3-9 所示。

表 3-9 legend 函数的常用参数

参数	说明
x 和 y	设置图例的位置（左上角位置） 除了使用 x 和 y 参数外，也可以使用“bottomright”、“bottom”、“bottomleft”、“left”、“topleft”、“top”、“topright”、“right”、“center”参数
legend	一个字符向量，表示图例中的文字
horiz	为 FALSE（默认）时，图例垂直排列 为 TRUE 时，图例水平排列
ncol	图例的列数目。如果 horiz 为 true，则此项无意义
pch	图例中点的样式。可以既设置 pch，又设置 lty，并可以设置为 NA，表示某组图例无点样式。 具体的取值含义可以参考 1.2.3 节中“元素相关参数设置”
lty	图例中线的样式。可以既设置 pch，又设置 lty，并可以设置为 NA，表示某组图例无线样式。 具体的取值含义可以参考 1.2.4 节“线元素相关参数设置”
col	图例中点 / 线的颜色
bg	图例的背景颜色。在 bty 参数为“n”时无效
bty	设置图例框的样式：默认为“o”，表示显示边框。 设置为“n”表示无边框。其他取值的意义可以参考 1.3.5 节“边框”
title	设定图例的标题

下面通过几个实例进行介绍。先来看如下的代码。

```
    text.legend=c("上周pv","本周pv","pv同比增长","pv环比增长")
    col2<-c("black","blue")
    legend("topleft",pch=c(15,15,16,16),legend=text.legend,col=c(col,col2),bty="n",
horiz=TRUE)
```

上述代码用于设置图例的左上角（topleft），并通过 col 参数把图例中的颜色和主图形中的各组数据颜色进行对应。由于只设置了 pch 参数，而未设置 lty 参数，所以图例中的 icon 图标全部使用点来代替。bty 为 n 时，图例无边框。horiz 设置为 TRUE 时，表示每组图例水平排列。最终的效果如图 3-18 所示。

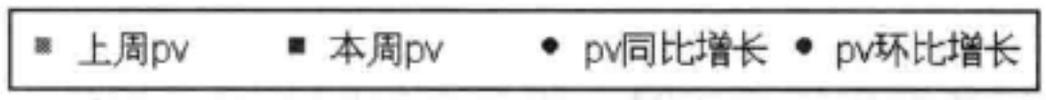


图 3-18 图例

下面这段代码不再使用特定字符串来设置位置，而是直接设置位置（x=1，y=500），位置仍在左上角。

```
legend(1,500,pch=c(15,15,NA,NA),lty=c(NA,NA,1,1),legend=text.legend,col=c(col,c
ol2),ncol=2,bg="aliceblue")
```

其中，pch 参数设置前两组图例是点样式（15），lty 参数设置后两组图例是线样式（1）。ncol=2，表示 4 组图例排列为 2 列。由于没有设置 bty 参数，所以该图例默认没有边框。通过 bg 背景颜色参数设置图例颜色为 aliceblue。最后的效果如图 3-19 所示。



图 3-19 图例

2. 直接在绘图函数中设置参数

直接在 barplot 函数设置 legend.text 参数图标的文字，系统会直接选取相应的颜色、线条及点样式。这种方式较为便捷，但是灵活性较差，笔者习惯使用 legend 函数设置的方式。

3.3.4 坐标轴

坐标轴的设置主要包括主坐标轴（x 轴和 y 轴）的范围和刻度标记，以及副坐标（右侧的纵坐标）的相关属性等。

1. axis 函数

axis 函数可以在上、下、左、右 4 个边上设置坐标轴，并设置坐标轴的范围 / 刻度标记等。其主要参数如表 3-10 所示。

表 3-10 axis 函数的主要参数

参数	说明
side	设置坐标轴所在的边。当取值为 1、2、3、4 时，分别表示坐标轴处于下、左、上、右各边
labels	通过向量来设置坐标轴内各刻度的名称（刻度标记）
font.axis	刻度标记的字体
cex.axis	刻度标记的大小
col.axis	刻度标记的颜色
at	通过向量来设置坐标轴内各刻度标记的位置。at 参数向量要与 labels 向量一一对应
tick	逻辑参数，如果 tick=TRUE（默认），则画出坐标轴。 如果 tick=FALSE，则不画出坐标轴。注意，此时并不影响刻度标记 labels 的展示
col	坐标轴的颜色。tick=TRUE 时，有效

（续）

参数	说明
col.ticks	坐标轴刻度线的颜色。 注意：col.ticks 是指与坐标轴垂直的小刻度线的颜色。 col 表示设置了除刻度标记（labels）以外的部分颜色，包括 col.ticks
lty	坐标轴的样式。tick=TRUE 时，有效。具体取值的含义可以参考 1.2.4“线元素相关参数设置”
lwd	坐标轴的宽度。tick=TRUE 时，有效。具体取值的含义可以参考 1.2.4“线元素相关参数设置”

参数中的颜色有 3 种：col.axis、col、col.ticks，具体的解释已经在表 3-10 中给出。这里再使用如下一段代码加以说明：

```
axis(3, c(2,5,8,11,14,17,20), labels = text.x, cex.axis=0.75,col.
axis="red",col="blue",col.ticks="green")
```

产生的坐标轴样式如图 3-20 所示。

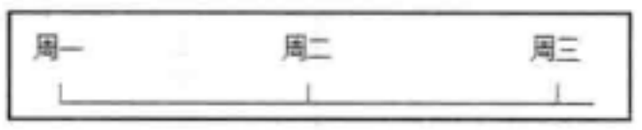


图 3-20 坐标轴样式

可以看到，刻度标记通过col.axis="red" 设置为红色。col="blue" 表示把其余部分设置为蓝色。最后通过 col.ticks="green" 语句把与坐标轴垂直的刻度线的颜色覆盖为绿色。

柱状图可用于分析数据在不同分类情况下的展示，例如，3.3.1 中的例子“本周的 pv 数据趋势”就采用了柱状图。柱状图的 x 轴不能单纯地用 1 ～ 7 表示，这并不能体现实际意义。相比之下，更加希望将 x 轴的相关刻度标记为：“周一”～“周日”。要实现此目标不难，继续加载如下代码：

```
text.x<-c("周一","周二","周三","周四","周五","周六","周日")
axis(side=1, c(2,5,8,11,14,17,20), labels = text.x, tick = FALSE, cex.axis =
0.75)
```

最后结果如图 3-21 所示。

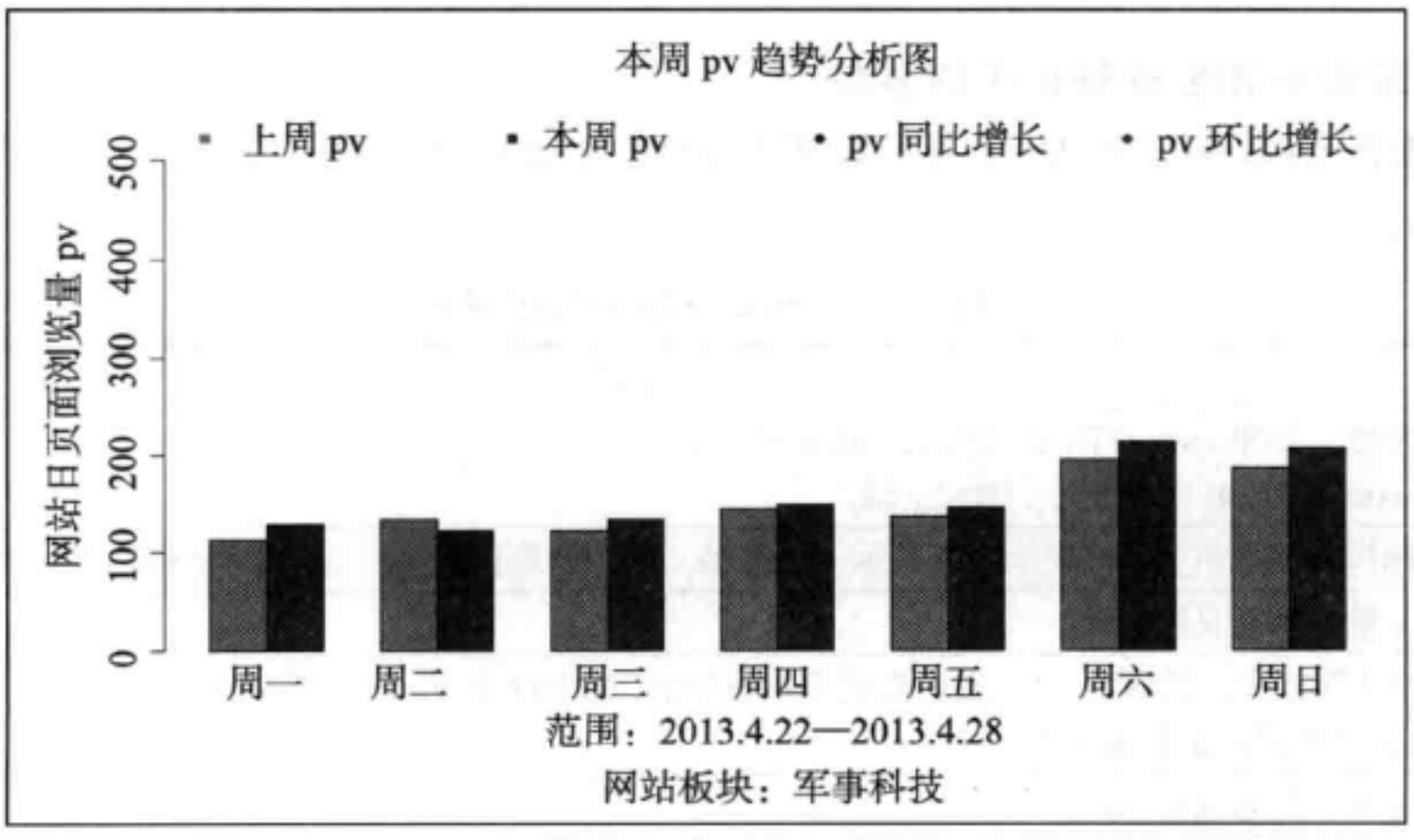


图 3-21 网站 PV 引例图 3

这里通过 labels = text.x，设置刻度标记为周一至周日，并设置 cex.axis=0.75，使标记文字缩小。通过 tick = FALSE 隐藏其余坐标轴及其刻度线的展示。通过设置 side 参数为 1，使该语句用于设置底部的 x 轴。设置 at 参数为 c（2,5,8,11,14,17,20），在相应的横坐标处显示刻度标记。

在 PV 趋势分析中，还希望在柱形图的上方画出 pv 的同比增长率和环比增长率。这就需要在右侧上方绘制一个副 y 坐标轴，其刻度范围是 −60% ～ 60%。代码如下：

```
    axis(4,at=seq(from=250,length.out=7,by=40),labels=c("-60%","-40%","-20%","0","2
0%","40%","60%"))
```

显示效果如图 3-22 所示。

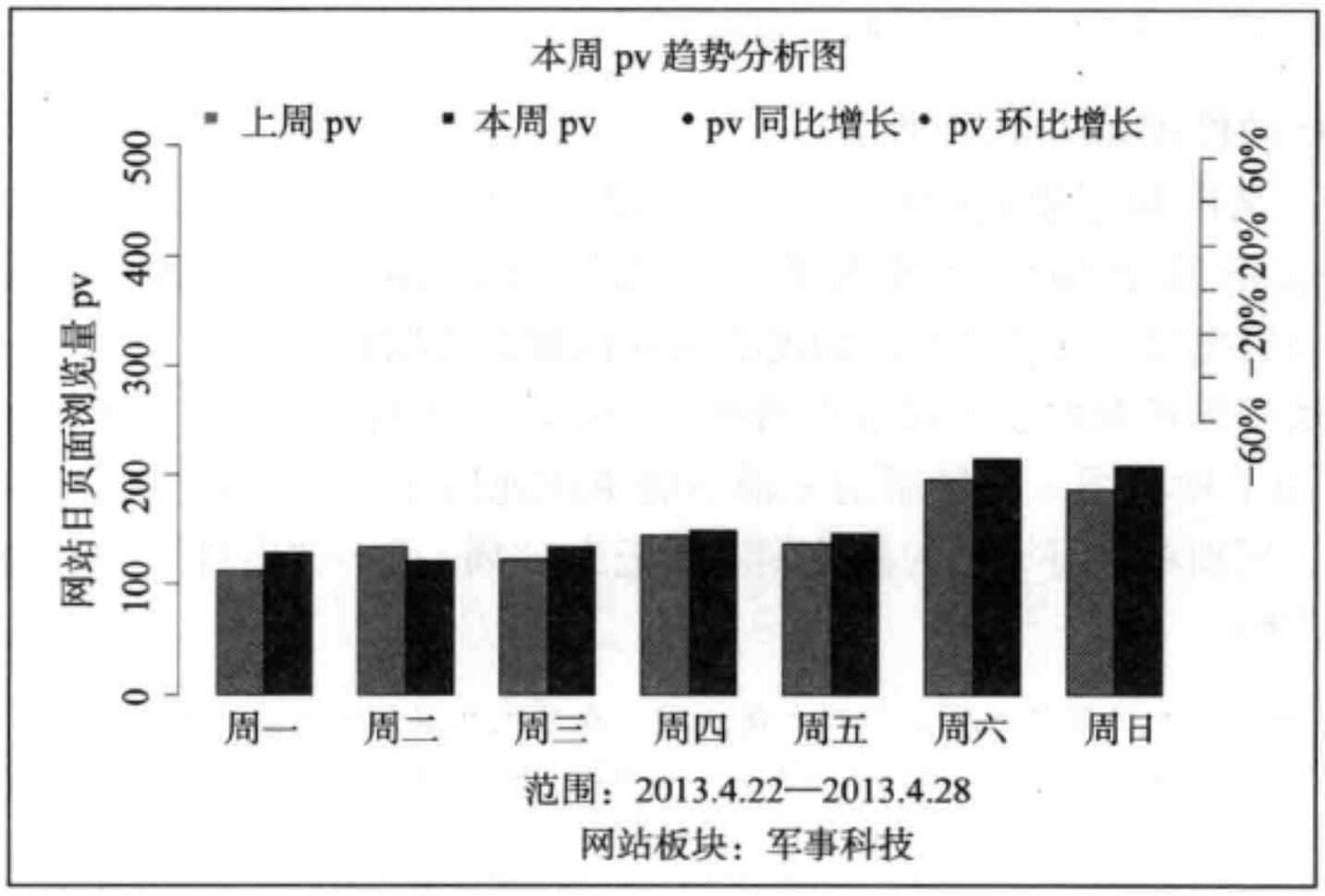


图 3-22 网站 PV 引例图 4

2. 绘图函数中和坐标轴相关的参数

在高级绘图函数中一般都有用于设置坐标轴的展示和范围的 axes、xlim 和 ylim 参数，如表 3-11 所示。

表 3-11 和坐标轴相关的参数

参数	说明
axes	逻辑参数，如果 axes=TRUE（默认），则显示 x 轴和 y 轴。 如果 axes=FALSE，则隐藏 x 轴和 y 轴
xaxt	坐标轴样式，取值为 "n" 表示隐藏显示 x 坐标轴。默认的取值是 "s"，表示 x 坐标轴以标准样式显示
yaxt	设置 y 轴，其意义同 xaxt
xaxs	x 轴的计算方式，默认为 "r"，表示先把原始数据的范围向外扩大 4%，然后用这个范围画坐标轴；取值为 "i" 表示直接使用原始数据范围
yaxs	设置 y 轴，其意义同 xaxs

（续）

参数	说明
xlog	设置 x 轴坐标是否取对数，默认为 FALSE
ylog	设置 y 轴，其意义同 xlog
xlim	x 轴的范围。设置为 c（from,to），from 是 x 轴的首坐标，to 是尾坐标
ylim	y 轴的范围

注意，如果要隐藏 x 轴和 y 轴，则设置 axes=FALSE。如果要隐藏 x 轴，只显示 y 轴，则设置 axes 参数为 FALSE，并在后续使用 axis（2）命令，也可以直接设置 xaxt 参数为“n”。再来回顾一下引例中的部分代码。

```
ylim.max<-550
col=c("azure4","brown4")
barplot(as.matrix(rbind(data$pre,data$now)),
        beside=TRUE,ylim=c(0,ylim.max),col=col,axes=F)
axis(2)
```

设置 axes 参数为 F 后，barplot 柱状图的 x 轴和 y 轴就都隐藏了。然后通过 axis（2）代码又显示了 y 轴的坐标，也可以直接使用 xaxt="n" 来隐藏 x 轴。ylim=c（0,ylim.max）设置 y 轴的范围是 0 ～ 500。

3. 坐标轴的密度分布展示

在绘制样本的散点图时，可以使用 rug 函数体现相应样本的分布情况，代码如下：

```
plot(conver,pv,main="转化率 - 单品页浏览量")
rug(conver)
rug(pv,side=2)
```

结果如图 3-23 所示。

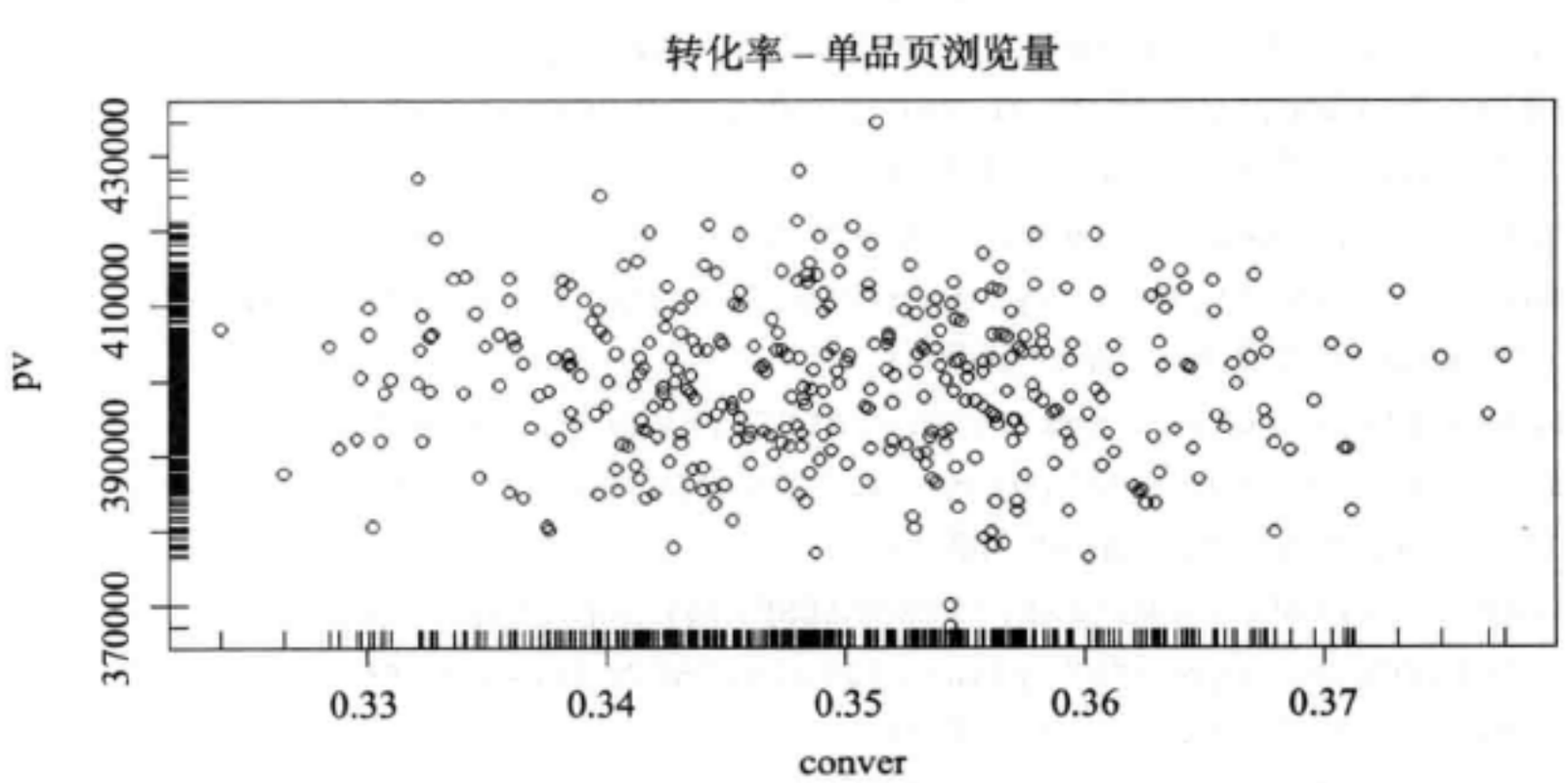


图 3-23 转化率 – 单品页浏览量的散点图

在图 3-23 中，x 轴是商品的转化率，使用 rug（conver）命令绘制转化率数据在 x 轴的分布情况。其中黑线越密集，表示转化率数据在该区域内的分布越集中。rug（pv,side=2）绘制 pv 数据在 y 轴的分布情况，注意 side=2 表示绘制位置在左侧 y 轴上。默认 side=1，表示底部 x 轴。

3.3.5 边框

1. 绘图函数的 bty 参数

bty 参数可设置图形边框样式，取值为字符 "o"、"l"、"7"、"c"、"u"、"]"；这些字符本身的形状对应边框样式。例如，o（默认值）表示 4 条边框都显示，c 表示不显示右边框。bty 为 "n" 时，表示不绘制任何边框。

注意

如果 axes 参数设置为 FALSE，则 bty 参数设置无效。

bty 取不同数值时的代码如下：

```
    x.text<-c("1月","2月","3月","4月","5月","6月","7月","8月","9月","10月","11月","12月")
    sales.volume<-c(158721,190094,108441,88092,68709,50116,90117,160044,186045,106334,89092,104933)
    par(mfrow=c(2,3))
    plot(sales.volume,type="b",ylim=c(20000,250000),xaxt="n",yaxt="n",main="bty默认取\"o\"",xlab="月份",ylab="销量")
    axis(1,at=1:12,labels=x.text,tick=FALSE);axis(2,tick=FALSE)
    plot(sales.volume,type="b",ylim=c(20000,250000),xaxt="n",yaxt="n",bty="l",main="bty取\"l\"",xlab="月份",ylab="销量")
    axis(1,at=1:12,labels=x.text,tick=FALSE);axis(2,tick=FALSE)
    plot(sales.volume,type="b",ylim=c(20000,250000),xaxt="n",yaxt="n",bty="7",main="bty取\"7\"",xlab="月份",ylab="销量")
    axis(1,at=1:12,labels=x.text,tick=FALSE);axis(2,tick=FALSE)
    plot(sales.volume,type="b",ylim=c(20000,250000),xaxt="n",yaxt="n",bty="c",main="bty取\"c\"",xlab="月份",ylab="销量")
    axis(1,at=1:12,labels=x.text,tick=FALSE);axis(2,tick=FALSE)
    plot(sales.volume,type="b",ylim=c(20000,250000),xaxt="n",yaxt="n",bty="u",main="bty取\"u\"",xlab="月份",ylab="销量")
    axis(1,at=1:12,labels=x.text,tick=FALSE);axis(2,tick=FALSE)
    plot(sales.volume,type="b",ylim=c(20000,250000),xaxt="n",yaxt="n",bty="]",main="bty取\"]\"",xlab="月份",ylab="销量")
    axis(1,at=1:12,labels=x.text,tick=FALSE);axis(2,tick=FALSE)
```

效果如图 3-24 所示。

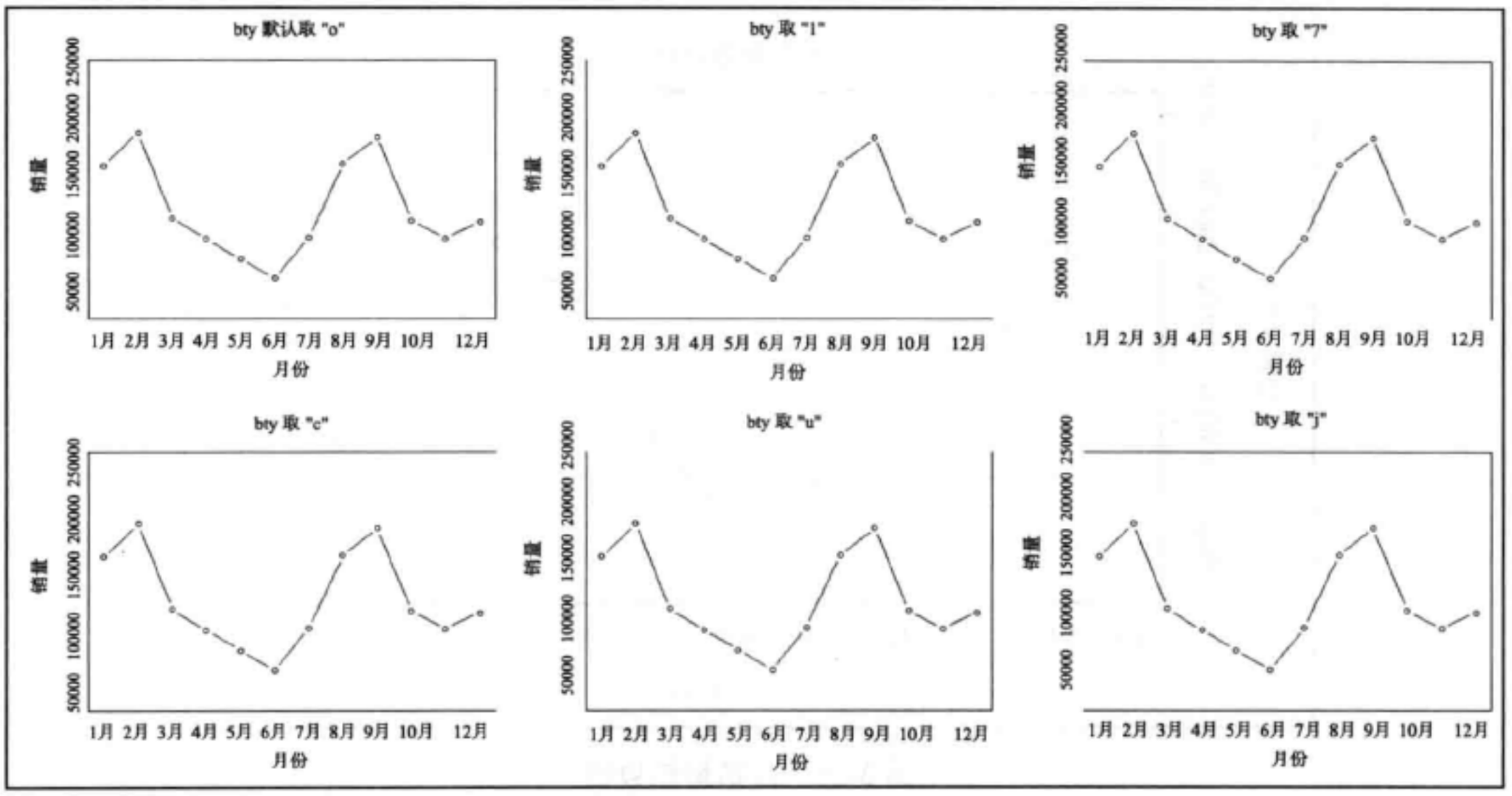


图 3-24 参数 bty 设置不同数值时的样式

2. box() 函数

通过 box() 函数也可以设置各边框的线条样式。其参数可为：

- ❑ bty：边框样式，其具体的取值意义在上文已经详细介绍了。
- ❑ col：边框颜色。
- ❑ lwd：边框线条的宽度。
- ❑ lty：边框线条的样式。

以上各参数均可以参考 3.2.4 “关线元素相关参数设置” 的内容。

3.3.6 网格线

使用 grid() 函数可以在绘图的基础上添加网格线，其参数主要包括：ny 用于设置水平网格的数目，nx 用于设置垂直网格的数据。设置为 NA 时，表示不绘制相应的网格线。例如：

```
    x.text<-c("1月","2月","3月","4月","5月","6月","7月","8月","9月","10月","11
月","12月")
    sales.volume<-c(158721,190094,108441,88092,68709,50116,90117,160044,186045,1063
34,89092,104933)
    plot(sales.volume,type="b",ylim=c(20000,250000),xaxt="n",main="月 销 量 趋 势 图
",xlab="月份",ylab="销量（元）")
    axis(1,at=1:12,labels=x.text,tick=FALSE);
    grid(nx=NA, ny=8,lwd=1,lty=2,col="blue")
```

效果如图 3-25 所示。

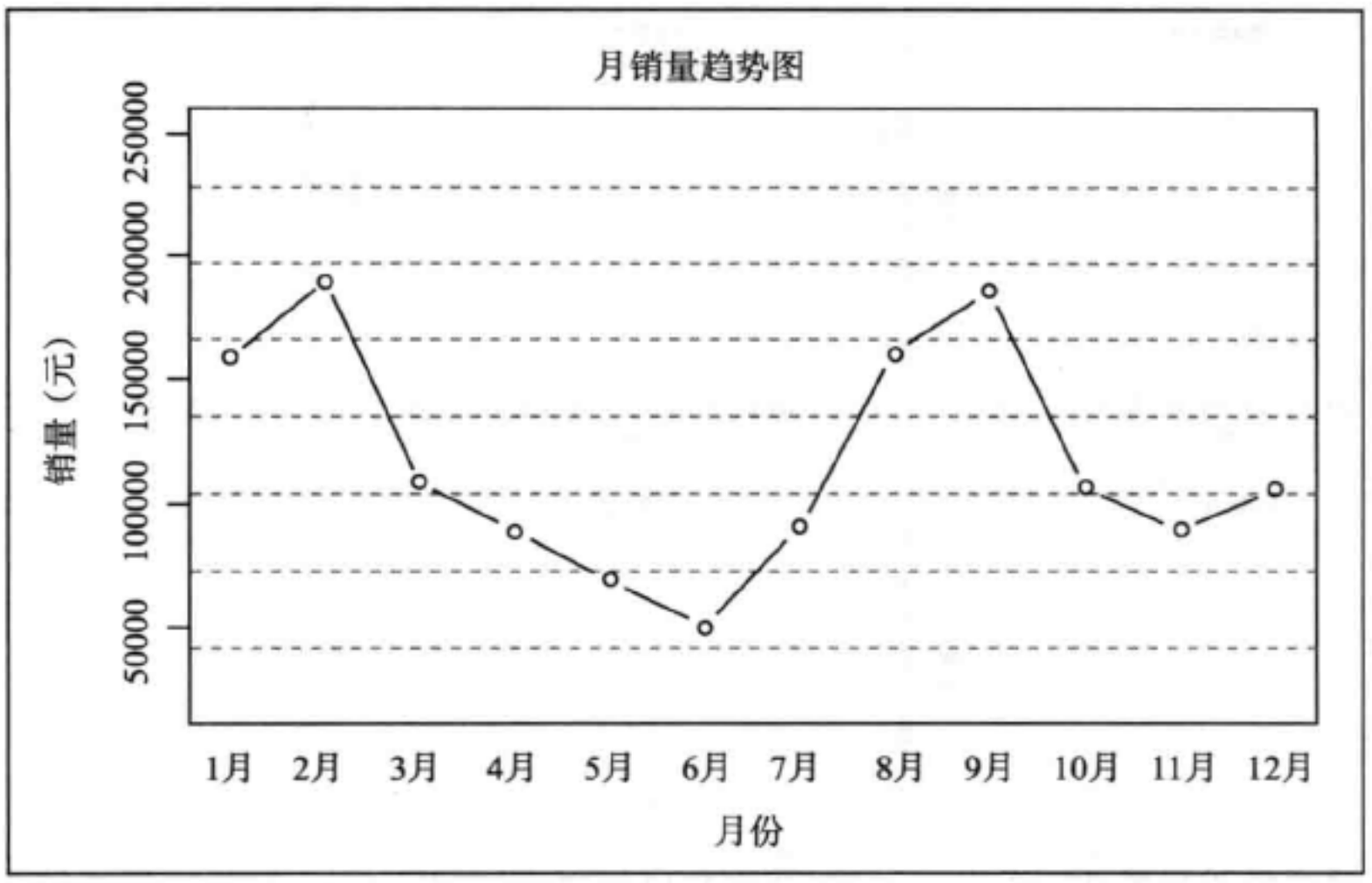


图 3-25 月销量趋势图

在图 3-25 中，nx 取 NA 表示不绘制垂直网格线，ny 取 8 表示绘制 8 条水平网格线。lwd、lty 和 col 参数分别设置网格线的宽度、样式和颜色，其具体含义可参考 3.2.4 “线元素相关参数设置”。

3.3.7 点

通过低级绘图函数 points 可以独立添加点元素。其主要参数如表 3-12 所示。

表 3-12 points 函数的主要参数

参数	说明
x	横坐标位置，可以设置向量来代表多个点的位置
y	纵坐标位置，可以设置向量来代表多个点的位置
type	有 9 种取值，分别代表不同的样式：'p' 表示画点（默认）；'l' 表示画线；'b' 表示同时画点和线，但点线不相交；'c' 表示将 type= 'b' 中的点去掉，只剩下相应的线条部分；'o' 表示同时画点和线，且相互重叠（这是与 type = 'b' 的区别）；'h' 表示画铅垂线；'s' 表示画阶梯线，从一点到下一点时，先画水平线，再画垂直线；'S' 也是表示画阶梯线，但从一点到下一点是先画垂直线，再画水平线；'n' 表示做一幅空图
pch	点样式，具体可参考 3.2.3 “点元素相关参数设置”
col	点颜色，具体可参考 3.2.3 “点元素相关参数设置”
bg	点的背景色，pch 取 21 ～ 25 时有效，具体可参考 3.2.3 “点元素相关参数设置”
lwd	点的边线的宽度，具体可参考 3.2.4 “线元素相关参数设置”
cex	点的大小，具体可参考 3.2.3 “点元素相关参数设置”

例如，绘制 type 的所有样式。代码如下：

```
x<-2:9;y<-2:9
par(mfrow=c(3,3))
```

```
    plot(1,ylim=c(1,10),xlim=c(1,10),col="white",main="type=\"p\" 默认 ");points(x,y)
    plot(1,ylim=c(1,10),xlim=c(1,10),col="white",main="type=\"l\"");points(x,y,type
="l")
    plot(1,ylim=c(1,10),xlim=c(1,10),col="white",main="type=\"b\"");points(x,y,type
="b")
    plot(1,ylim=c(1,10),xlim=c(1,10),col="white",main="type=\"c\"");points(x,y,type
="c")
    plot(1,ylim=c(1,10),xlim=c(1,10),col="white",main="type=\"o\"");points(x,y,type
="o")
    plot(1,ylim=c(1,10),xlim=c(1,10),col="white",main="type=\"h\"");points(x,y,type
="h")
    plot(1,ylim=c(1,10),xlim=c(1,10),col="white",main="type=\"s\"");points(x,y,type
="s")
    plot(1,ylim=c(1,10),xlim=c(1,10),col="white",main="type=\"S\"");points(x,y,type
="S")
    plot(1,ylim=c(1,10),xlim=c(1,10),col="white",main="type=\"n\"");points(x,y,type
="n")
```

效果如图 3-26 所示。

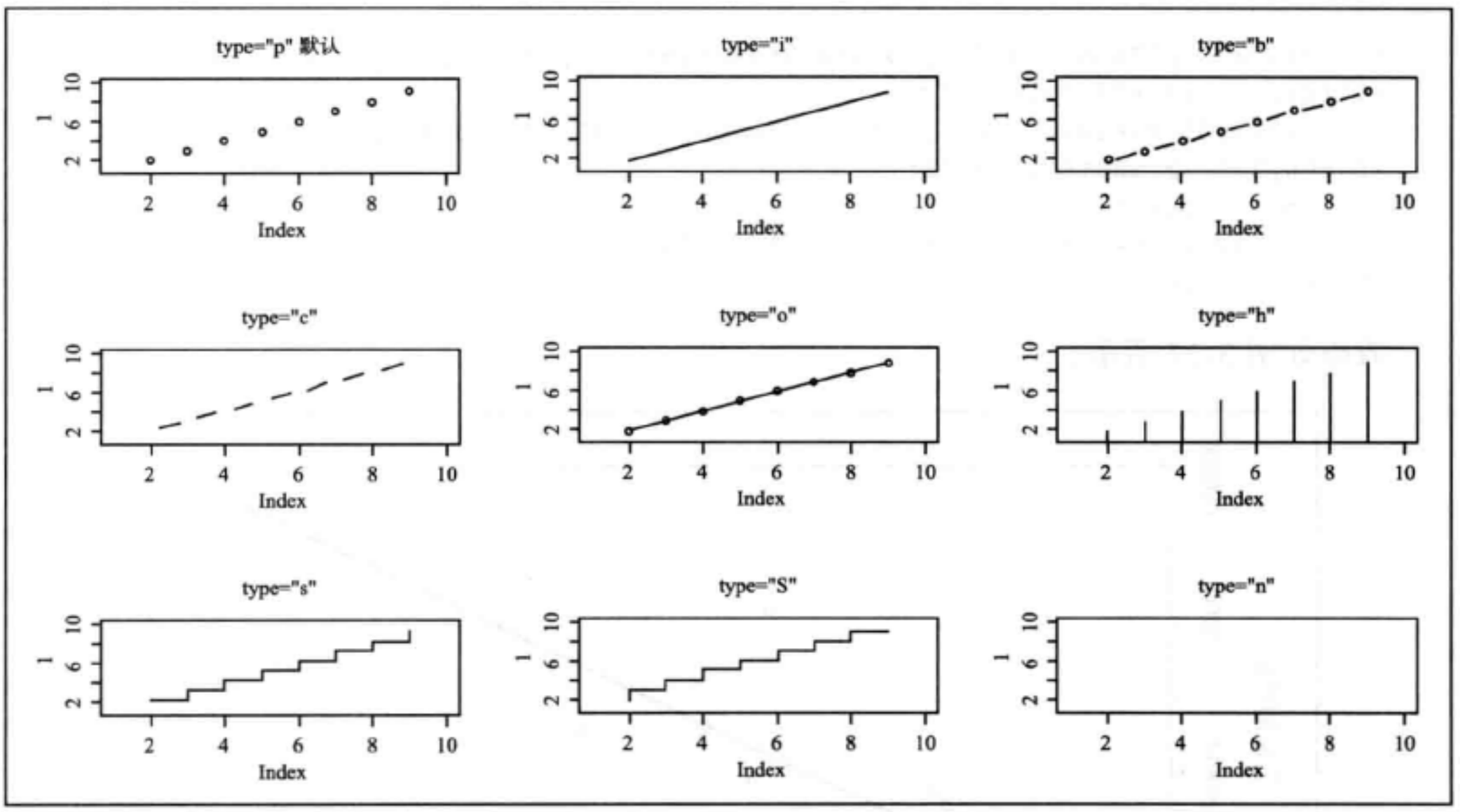


图 3-26 设置不同 type 参数值时的样式

3.3.8 线

从 3.3.7 的内容可知，type 的值为 l、b、c、o 时，实质已表示通过一组向量的点将点画成了线，下面介绍如何使用函数来绘制曲线（lines 函数）、直线（abline 函数）、线段（segments 函数）的介绍。

1. 使用 abline() 函数绘制直线

abline 函数的基本形式是：

abline（a = NULL, b = NULL, h = NULL, v = NULL,coef = NULL, ...）

其中，a 是截距，b 是斜率，h 是画水平线时的纵轴值，v 是画垂直线时的横轴值。coef 是一个能用函数 coef() 提取系数（包含斜率和截距）的 R 对象，典型的使用案例就是用线性模型（回归）生成的对象，其系数是一个长度为 2 的向量，并分别表示为直线的截距和斜率。abline 函数的应用实例如表 3-13 所示。

表 3-13 abline 的应用实例

类型	形式	说明
直线	abline（a,b）	画出 y=a+bx 直线
水平线	abline（h=y）	画出一条由所有纵坐标等于 y 值的点所组成的水平直线
垂直线	abline（v=x）	画出一条由所有横坐标等于 x 值的点所组成的水平直线
线性模型回归直线	abline（lm.obj）	画出由 lm 函数获取的回归直线的模型

具体实例如下：

```
pv<-c(532251,574540,537528,593018,759865,843321,785337,637756,627096,618487,
644126,535169,677159,635077)
uv<-c(223065,216166,217605,226722,258608,319029,296505,239219,237991,247237,
260646,233432,280163,269671)
sol.lm<-lm(pv~uv)
plot(pv~uv,xlab=" 独立用户数 ",ylab=" 页面浏览量 ")
abline(sol.lm)
```

效果如图 3-27 所示。

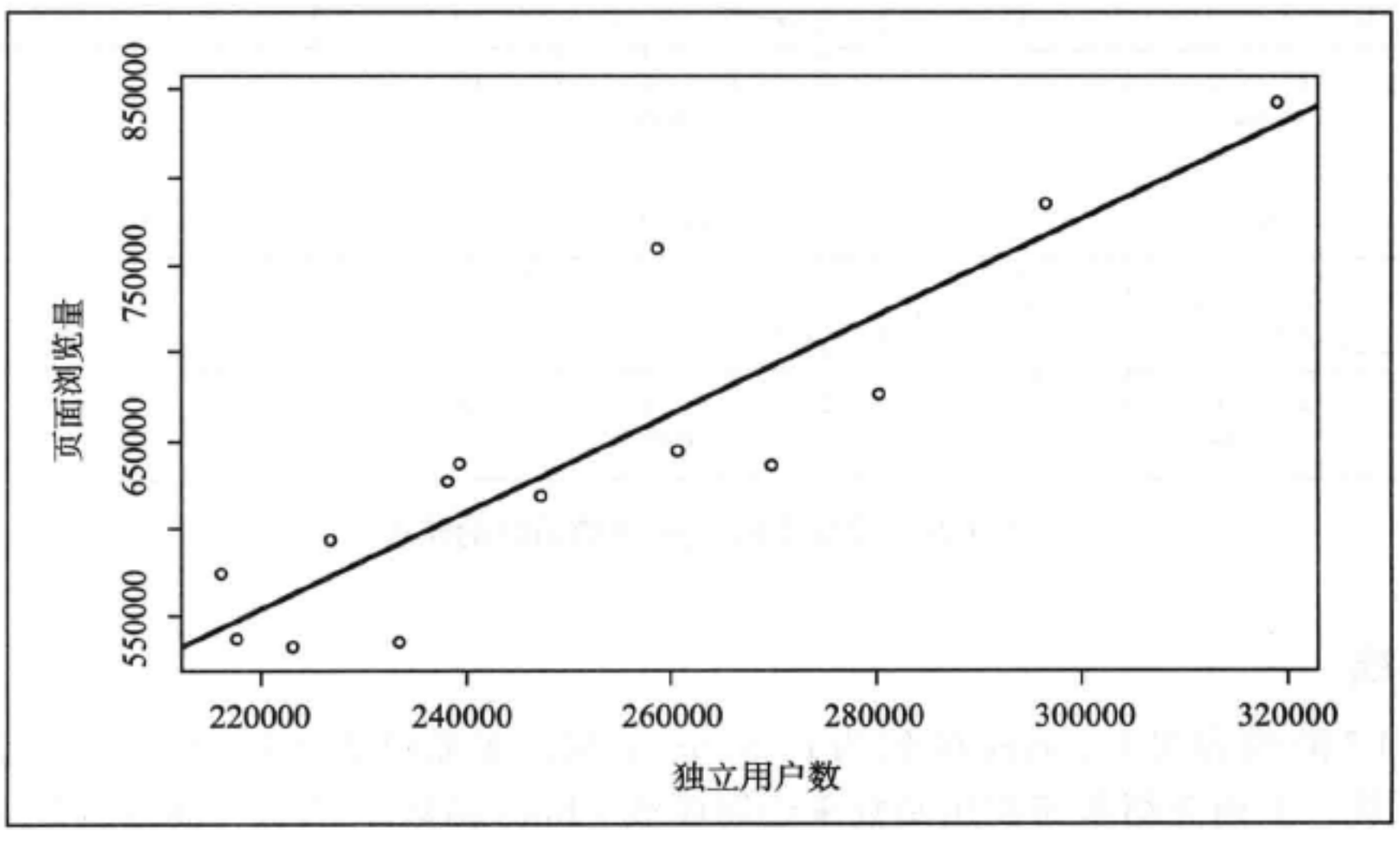


图 3-27 独立用户数—页面浏览量线性回归模型

这其中的 lm 是回归模型函数，这会在后面的章节详细讲解。lm（pv ～ uv）表示最后得出的模型是一个线性模型。pv=b+a*uv，sol.lm 包括回归线的斜率和截距。

abline 函数还有 col、lty、lwd 等关于线元素的参数。其具体取值可以参考 3.2.4 “线元素相关参数设置”。

2. 使用 segments() 函数绘制线段

和 abline 不同，segments 函数只绘制一段直线，而并非绘制贯穿整个绘图窗口的直线。在 segments（x0, y0, x1 = x0, y1 = y0, ...）中，主要的 4 个参数代表线段的起点和终点坐标。此外还有 col、lty、lwd 等关于线元素的参数，其具体取值可以参考 3.2.4 “线元素相关参数设置”。

3. 使用 arrows() 函数绘制箭头

arrows 函数与 segments 函数很类似，只是多了一个 angle 参数用于设置箭头样式，它表示箭头尖短线的角度（默认为 30 度）。

例如，绘制 angle 参数分别为 90、60、30、0 时的样式。代码如下：

```
plot(1,xlim=c(0,10),ylim=c(0,8),col="white")
arrows(1, 1, 8, 1, angle = 90);text(9,1,"angle=90")
arrows(1, 3, 8, 3, angle = 60);text(9,3,"angle=60")
arrows(1, 5, 8, 5, angle = 30);text(9,5,"angle=30")
arrows(1, 7, 8, 7, angle = 0);text(9,7,"angle=0")
```

效果如图 3-28 所示。

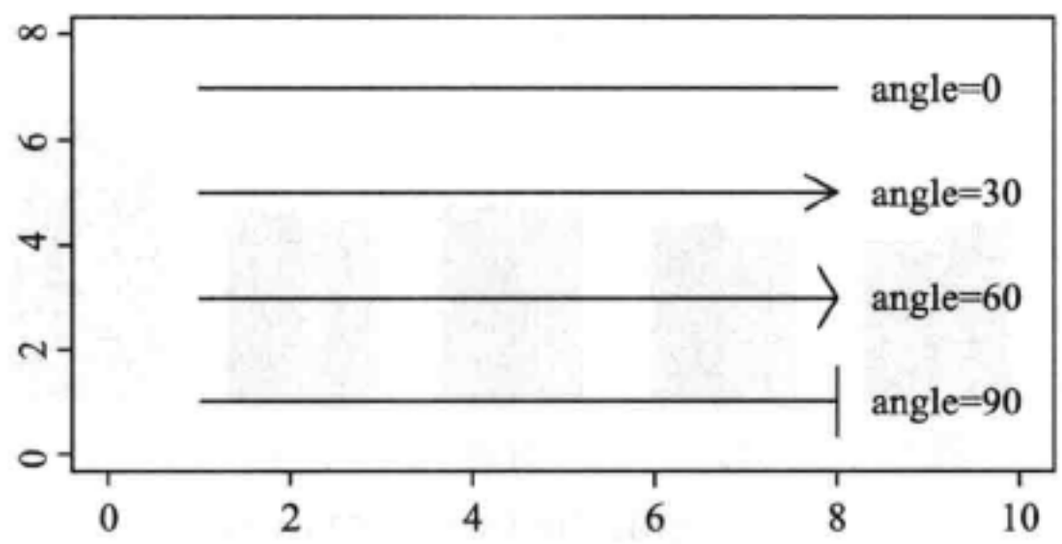


图 3-28　设置不同 angle 参数值时的样式

4. lines() 函数绘制曲线

前面介绍的 3 个函数均用于绘制直线，lines 函数则更加灵活，可通过散点样本的坐标绘制任意曲线。plot 是绘制散点图的高级绘图函数，但是它并没有 add 参数，即在同一绘图窗口中不能使用两个 plot 函数，此时可以使用 lines 函数。

在 lines（x, y = NULL, type = "l", ...）中，x 和 y 分别是散点样本的 x 轴与 y 轴坐标数据。type 参数可以参考 3.3.7 中 points 函数中的相关说明。对于 col、lty、lwd 等关于线元素的参数，可以参考 3.2.4 “线元素相关参数设置”。

下面将在图 3-22 的基础上，添加同比和环比增长率曲线。代码如下：

```
# 同比增长 =(now[t]-pre[t])/pre[t]
same.per.growth<-(data$now-data$pre)/data$pre
# 环比增长 =(now[t]-now[t-1])/now[t-1]
ring.growth<-c(NA,diff(data$now)/data$now[1:(length(data$now)-1)])
a<-200;b<-370
lines(c(2,5,8,11,14,17,20),a*same.per.growth+b,type="b",lwd=2)
lines(c(2,5,8,11,14,17,20),a*ring.growth+b,type="b",lwd=2,col="blue")
```

由于同比增长率和环比增长率的范围是 −100% ～ 100%，主坐标轴是 0 ～ 500，因此直接填在图形中时只是一条直线。所有要进行坐标转化，新坐标 =a* 旧坐标 +b。依据图 3-22 来调整，希望同比增长率为 60% 时，转化后的坐标为 490；同比增长率为 −60% 时，转化后的坐标为 250。即：

a × 0.6+b=490

a ×（−0.6）+b=250

最终解为 a=200，b=370。最后通过 lines 函数来绘制同比和环比的曲线图，如图 3-29 所示。

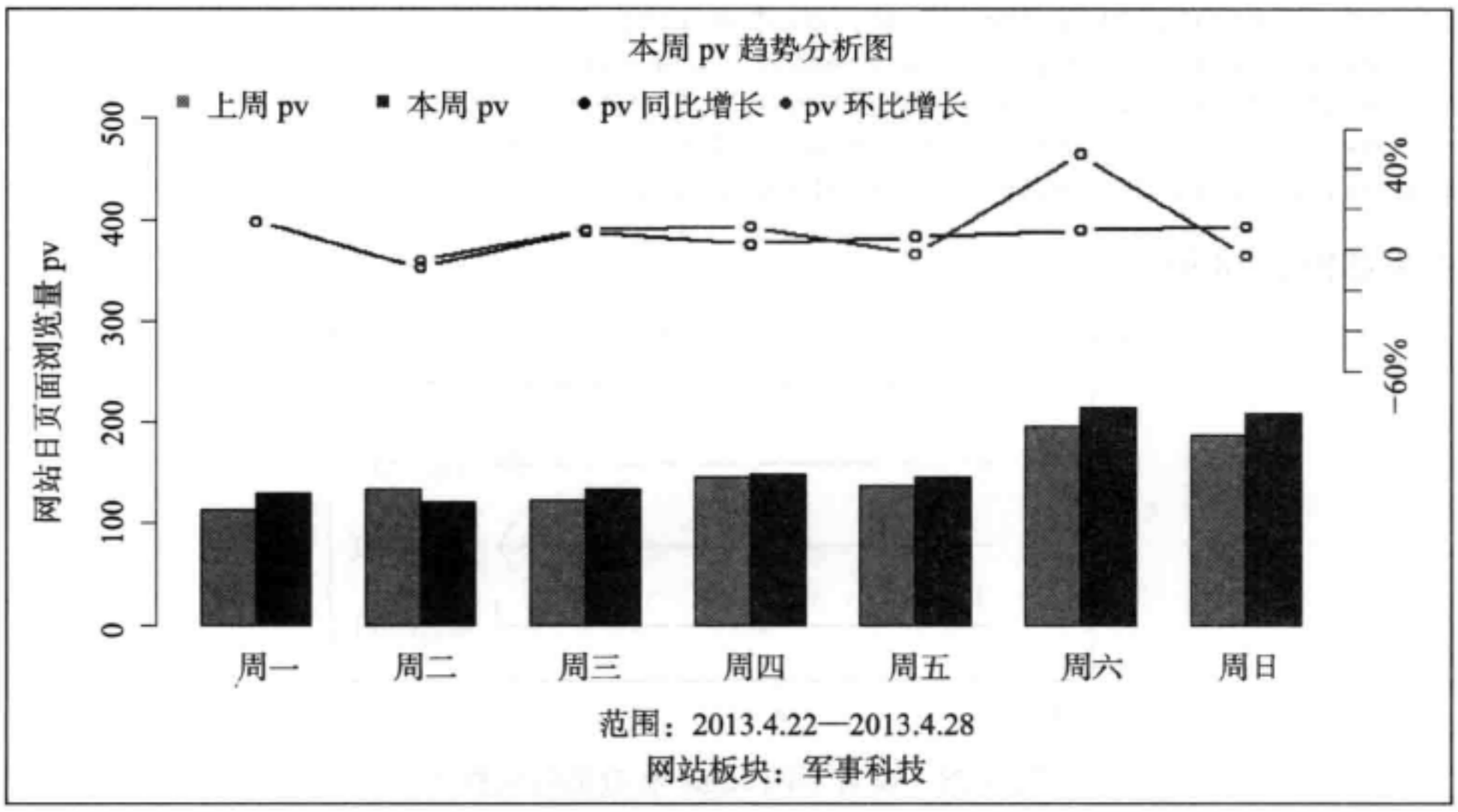


图 3-29 网站 PV 引例图 5

3.3.9 文字

text 函数用于在相应的位置添加文字，在 text(x, y = NULL, labels = seq_along(x),cex = 1, col = NULL, font = NULL, ...）中，x 和 y 表示绘制文字的位置向量；labels 用于设置文字内容；cex、col 和 font 参数分别表示文字的大小、颜色和字体，具体可以参考 3.2.2 “文字元素相关参数设置”。

下面在图 3-29 的基础上添加同比环比曲线，以及 PV 柱状图的数据文字。代码如下：

```
    # 在同比和环比曲线上添加文字
    j<-1
    for(i in 1:length(data[,1])){# 在 bar 内画出数据值
          text(3*i-1,a*same.per.growth[i]+b-5,paste(round(same.per.growth[i]*10000)
/100,"%",sep=""));j<-j+1
          text(3*i-1,a*ring.growth[i]+b+5,paste(round(ring.growth[i]*10000)/100,"%"
,sep=""),col="blue");j<-j+2
    }
    # 在 pv 柱状图上添加文字
    j<-1
    for(i in 1:length(data[,1])){# 在 bar 内画出数据值
        text(j+0.5,data$pre[i]+10,data$pre[i],col="deepskyblue4");j<-j+1
        text(j+0.5,data$now[i]+10,data$now[i],col="deepskyblue4");j<-j+2
}
```

效果如图 3-30 所示。

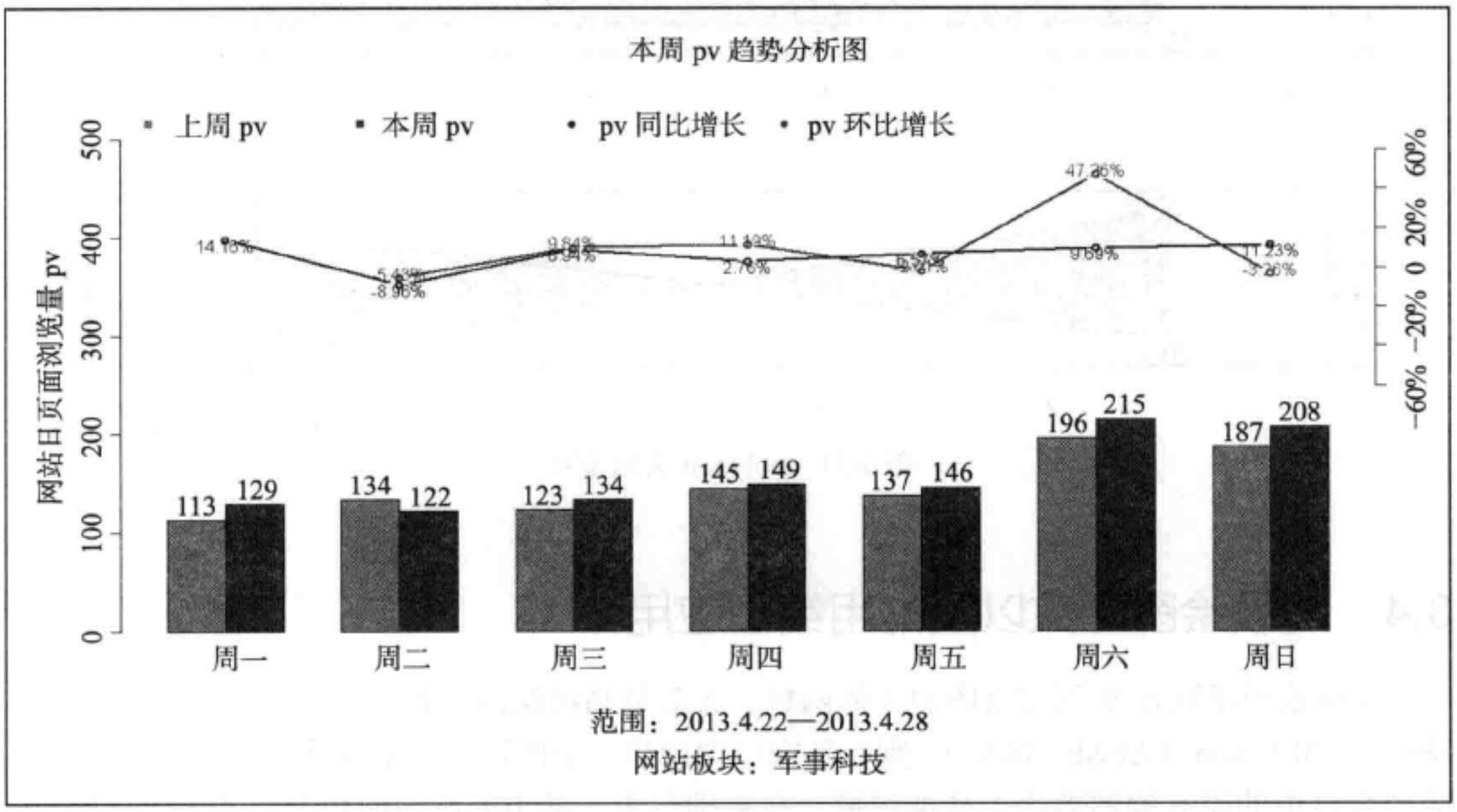


图 3-30 网站 PV 引例图 6

其中，paste（round（same.per.growth[i]*10000）/100,"%",sep=""）把“同比增长率”转化为“<同比增长率>%”形式的字符串。例如，same.per.growth[1] 是实数 0.14159292，转化后变为字符串 "14.15%"。

3.3.10 多边形

polygon 函数用于绘制一个多边形，其基本形式为：

polygon（x, y ,⋯ ）

其中，x 向量和 y 向量用于设置多边形各个顶点的坐标位置。例如：

```
par(mfrow=c(2,1))
plot(0,xlim=c(0,10),ylim=c(0,10),col="white")
polygon(x=c(1,1,9,9),y=c(9,1,1,9),col="orange",border="orange")
plot(0,xlim=c(0,10),ylim=c(0,10),col="white")
polygon(x=c(1,1,9,9),y=c(9,1,9,1),col="orange",border="orange")
```

注意，两个多边形的 4 个顶点坐标值相同，但是顶点的绘制顺序不同，这最终也会导致绘制的多边形不同，如图 3-31 所示。

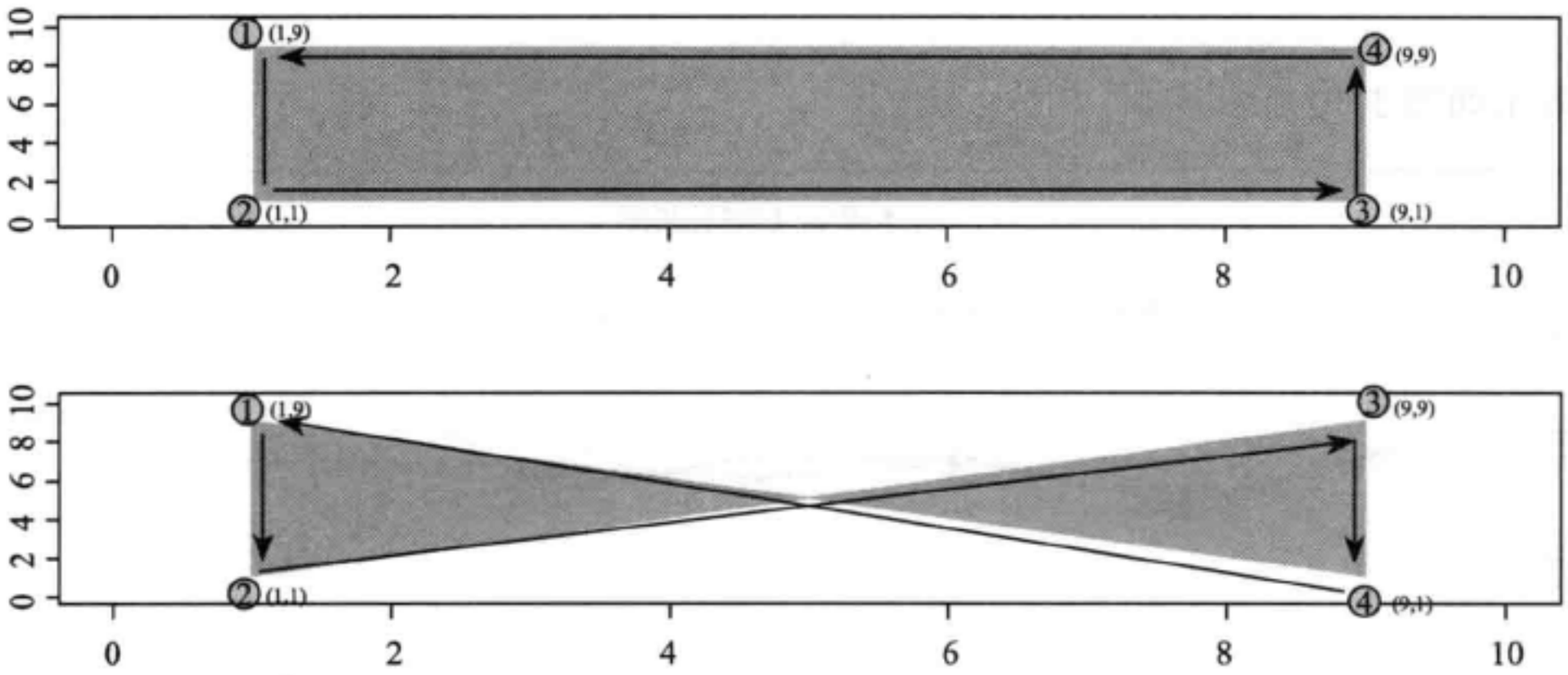


图 3-31 polygon 实例说明

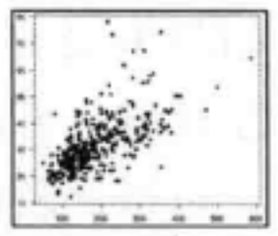
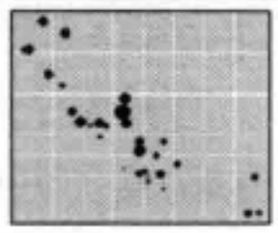
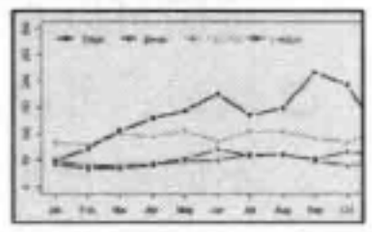
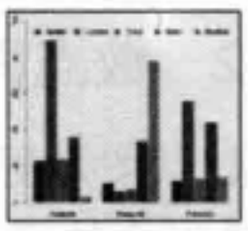
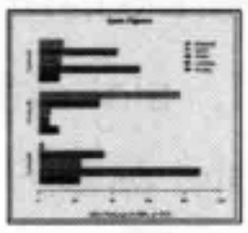
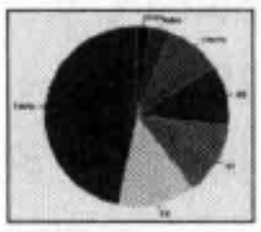
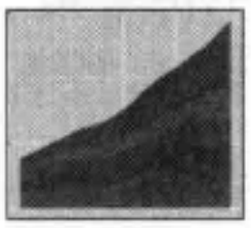
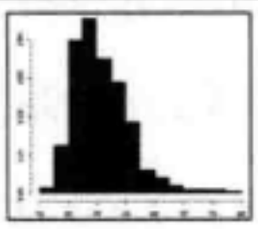
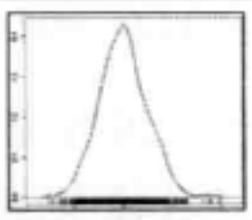
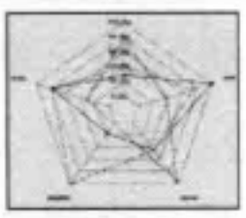
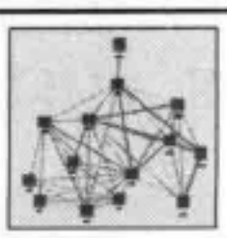
3.4 高级绘图函数以及常用绘图应用

高级绘图函数是 R 语言绘图的主体函数，大部分高级绘图函数均有 add 参数（plot 函数没有）。如果 add=FALSE（默认），则在新窗口中创建一个图形。如果 add=TRUE，则在当前活动窗口中的原有的图形之上叠加图形。在基础包中，常用的高级函数有：plot（绘制散点 / 曲线图）、barplot（绘制柱形 / 条形图）、hist（绘制直方图）、pie（绘制饼图）等。此外使用 ggplot2 扩展包可以绘制星状图、堆积面积图、气泡图等。

3.4.1 图形类型的选择

现在越来越多的人关注数据分析的结果展示，而良好的展示首先要选择适当的图形类型。表 3-14 中是业界常用的图形类型。

表 3-14 展示数据分析结果的常用图形

类型	说明	R 函数	图形示例
散点图	把一维数据或二维数据描绘在 x-y 二维平面上	plot	
气泡图	在散点图的基础上，使用点的大小来代表第三维数据	plot 或 ggplot2 包的 ggplot 函数	
曲线图	把散点图的所有样本点连接成线来展示	plot 和 lines	
柱形图	把主体数据通过垂直于 x 轴的柱形来展示	barplot	
条形图	柱形图翻转 90 度	barplot	
饼图	静态展示各个成分占整体的情况	pie	
堆积面积图	动态展示各个成分占整体的情况	ggplot2 包：qplot 函数	
直方图	分析一维数据的频次。分隔尺度是有限数据	hist	
密度曲线图	分析一维数据的频次。分隔尺度是无限小	plot（density()）	
雷达图 / 星状图	展示样本点的不同维度取值	fms 包的 radarchart 函数	
关系网络图	绘制不同样本点之间的联系程度	igraph 包：plot 函数	

对于各个图形的选择，本章将在国外专家 Andrew Abela 整理分析的基础上结合互联网常用分析方式，从比较、分布、构成、联系等 4 个大的方面入手，给出相应的图形选择思路，如图 3-32 所示。

- 比较：可以比较相同变量在不同时间下的数据，即所谓的时间序列分析和预测。同时可以比较相同变量不同分类下的数据。
- 分布：在分析变量数据前，往往要查看样本点的分布是否符合正态分布，以及是否含有异常点等。
- 联系：可以绘制散点图和气泡图，用以分析二维或者三维数据间的关系。使用网络关系图可以查看不同样本间的联系。
- 构成：可以静态比较各个成分占总体的比例，从而分清问题的主次。也可以动态查看不同成分的比例变化。

3.4.2 散点图

散点图是把样本的二维数据直接映射至 x 轴和 y 轴（如果是一维数据，则映射至 y 轴，x 轴是样本的序列号）的图形，其绘制函数是 plot。

1. plot 函数

plot 函数的基本形式为：

plot（x, y, ...）

其中 x 和 y 是样本的两个变量数据。如果样本变量只有一个，则需要把 x 参数设置为样本数据。除了设置 x 和 y 向量数据外，还可以设置 type 参数（具体可以参考 3.3.7 中“点元素的 type 参数”）、点以及线元素的相关属性参数（参考 3.2 节和 3.3 节的相关内容）。

例 1：绘制一维变量的样本点，y 轴显示 data 变量数值，x 轴显示样本序列号。

```
data<-c(11:20);
plot(data)
```

例 2：绘制二维变量的样本点，x 轴显示 x.data 变量数值，y 轴显示 y.data 变量数值。

```
x.data<-c(11:20);y.data<-c(20:11)
plot(x.data,y.data)
```

plot 函数除了可以绘制散点图之外，还可以通过设置 type 参数把散点串联起来变成曲线图（具体参考 3.4.4 的说明），设置 cex 参数控制点的大小，进而表现第三维变量的数值，并绘制包括三维数据变量信息的气泡图，具体可参考 3.4.3 的相关内容。

2. 实例：转化率 – 页面浏览量的散点图

下面绘制某网站首饰箱包类部分商品转化率（conver）和单品页日浏览量（PV）数据的散点图。具体数据如下：

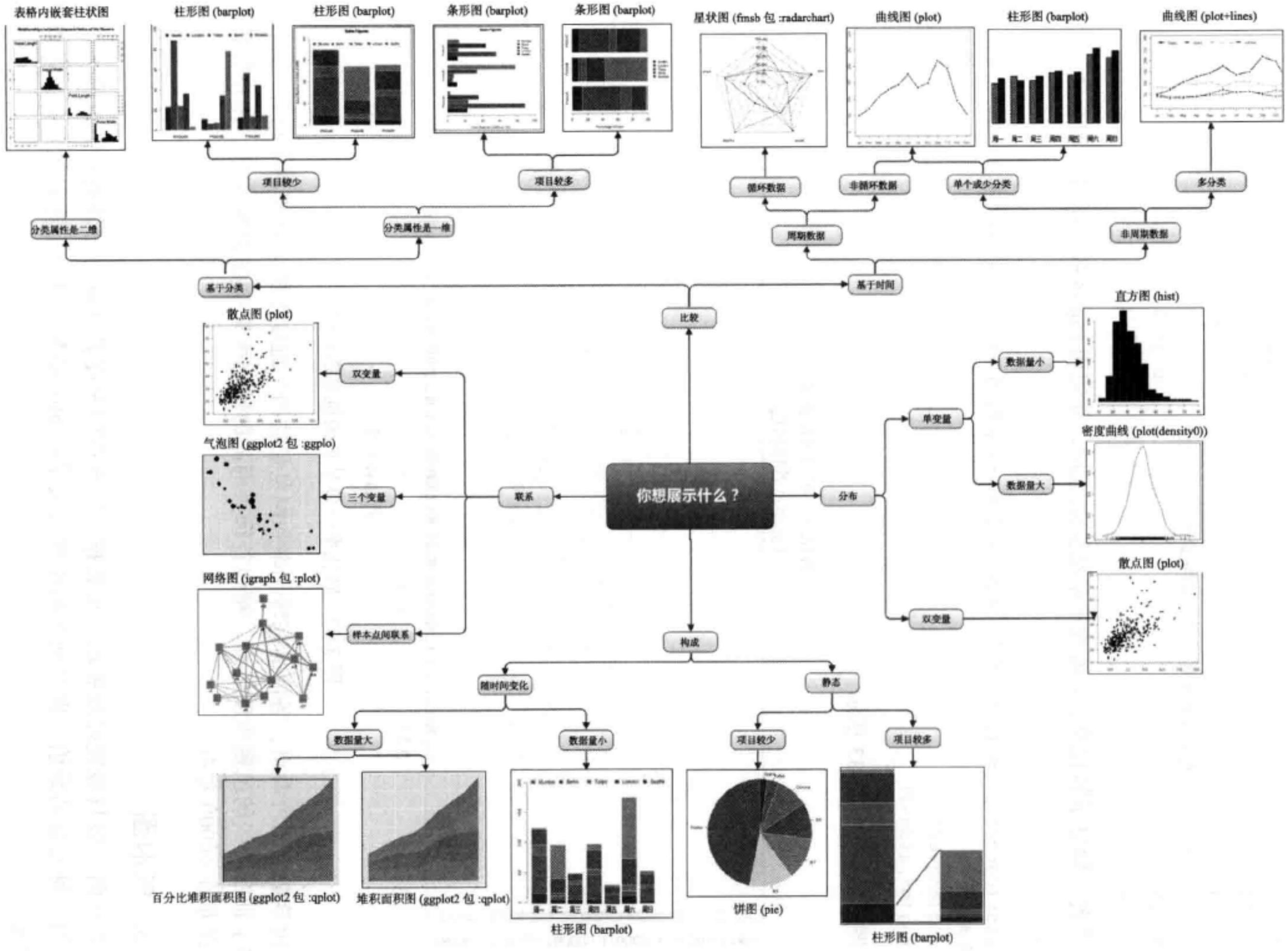

你想展示什么？
比较
分布
联系
构成
基于分类
基于时间
分类属性是二维
分类属性是一维
项目较少
项目较多
循环数据
非循环数据
单个或少分类
多分类
周期数据
非周期数据
单变量
双变量
数据量小
数据量大
随时间变化
静态
表格内嵌套柱状图
柱形图 (barplot)
柱形图 (barplot)
条形图 (barplot)
条形图 (barplot)
星状图 (fmsb 包 :radarchart)
曲线图 (plot)
柱形图 (barplot)
曲线图 (plot+lines)
散点图 (plot)
气泡图 (ggplot2 包 :ggplo)
网络图 (igraph 包 :plot)
直方图 (hist)
密度曲线 (plot(density0))
散点图 (plot)
百分比堆积面积图 (ggplot2 包 :qplot)
堆积面积图 (ggplot2 包 :qplot)
柱形图 (barplot)
饼图 (pie)
柱形图 (barplot)

图 3-32　图形选择思路

```
pv                                                    conver
406537                                                0.3550
400141                                                0.3461
384540                                                0.3497
        ……<此处由于数据过多，省略显示>……
425198                                                0.3534
400560                                                0.3510
```

设置x轴显示转化率，y轴展示单品页浏览量，并使用rug函数显示各轴的密度。代码如下：

```
plot(conver,pv,main="转化率 - 单品页浏览量 \n(品类: 首饰箱包)",xlab="商品转化率",ylab="单品页浏览量")
rug(conver)
rug(pv,side=2)
```

图形效果如图3-33所示。

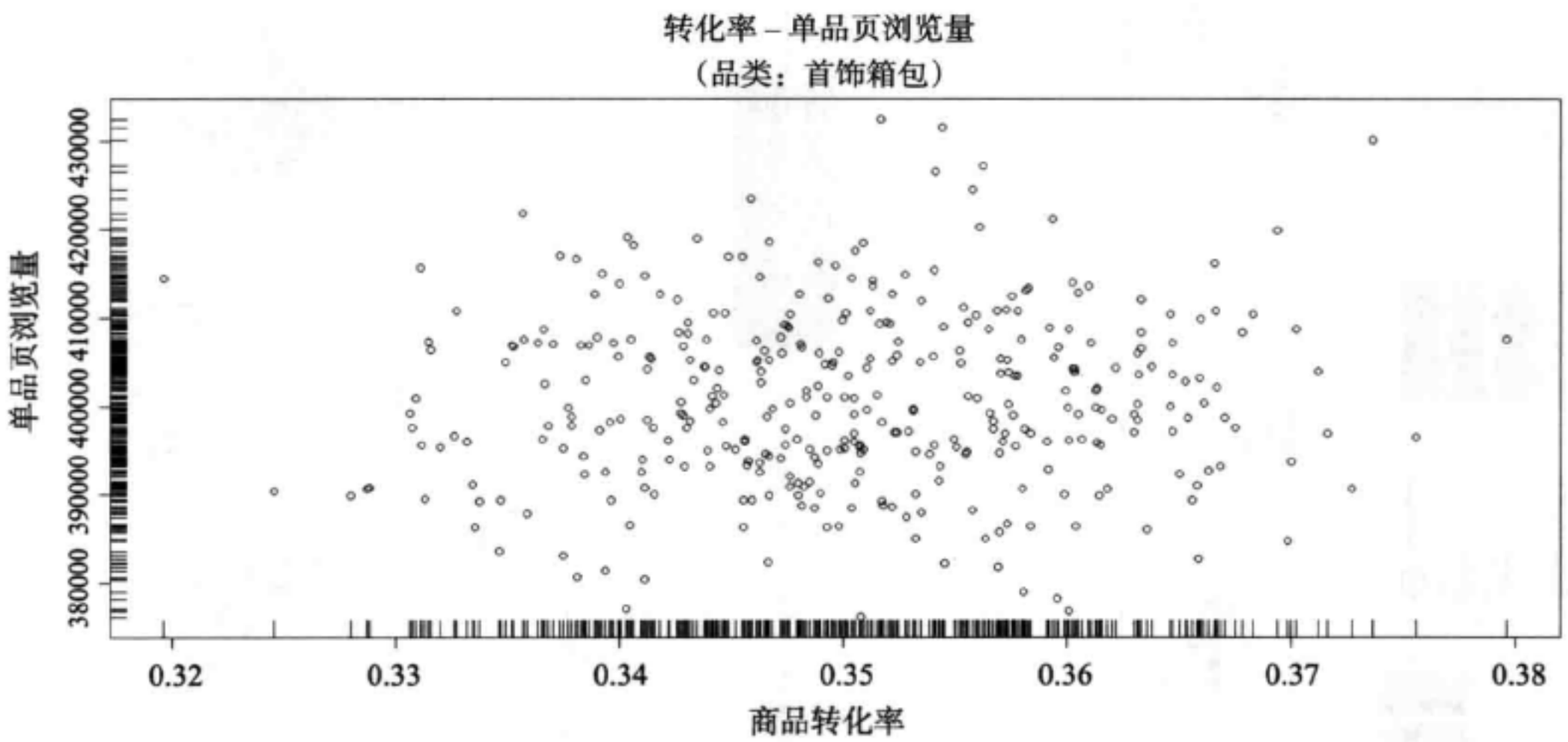


图3-33 转化率—单品页浏览量散点图

由该散点图可看出，各商品在转化率和浏览量这两个维度下的差异和集中趋势。观察x轴和y轴密度线的疏密程度可知，箱包首饰类商品的转化率集中在35%左右，用户每日浏览单品量在40000左右。

3.4.3 气泡图

散点图一般只能较好地展示二维数据，气泡图则是在其基础上另外通过散点的大小来表达第三维变量的数值。即气泡图的绘制方法是在plot函数的基础上通过设置cex参数来完成的。

实例：各类目转化率 – 页面浏览量 – 客单价的气泡图

以下是手机数码、食品饮料等 10 个大商品的类目转化率、页面浏览量，以及客单价的数据。

各类目转化率—页面浏览量—客单价数据

商品类目	转化率	页面浏览量（万）	客单价（元）
手机数码	0.012	23.19	3509
食品饮料	0.020	10.89	59
电脑办公	0.015	15.09	2501
家居用品	0.014	12.11	509
母婴玩具	0.018	9.60	411
家用电器	0.013	20.29	3011
服饰鞋帽	0.010	40.56	476
日用百货	0.025	28.66	81
虚拟商品	0.045	20.43	379
箱包礼品	0.011	13.84	610

现在绘制各个变量的散点图，其中 x 轴展示浏览量，y 轴展示转化率，散点的大小表示客单价。代码如下：

```
id<-c("手机数码","食品饮料","电脑办公","家居用品","母婴玩具","家用电器","服饰鞋帽","日用百货","虚拟商品","箱包礼品")
conver<-c(0.012,0.020,0.015,0.014,0.018,0.013,0.010,0.025,0.045,0.011)
pv<-c(23.19,10.89,15.09,12.11,9.60,20.29,40.56,28.66,20.43,13.84)
price<-c(3509,59,2501,509,411,3011,476,81,379,610) ←——— ①准备数据

library(RColorBrewer)
col<-brewer.pal(11, "Spectral")[2:11] ←——— ②设置不同类目的颜色

cex.max<-12# 最终显示的最大 cex
cex.min<-3# 最终显示的最小 cex
a<-(cex.max-cex.min)/(max(price)-min(price))
b<-cex.min-a*min(price)
cex2<-a*price+b ←——— ③设置散点大小

plot(pv,conver,col=col,cex=cex2,pch=16,ylim=c(0,0.06),xlab="页面浏览量（万）",ylab="转化率",main=list("各类目转化率 - 页面浏览量 - 客单价",cex=2),yaxt="n")
legend("topleft",legend=id,pch=16,col=col,bty="n",cex=0.8,ncol=5)
axis(2,labels=paste(seq(0,5,1),"%",sep=""),at=seq(0,0.05,0.01))
text(x=pv,y=conver,labels=price,cex=0.8)
text(x=40,y=0.055,labels="Z- 客单价",cex=1.3) ←——— ④绘制气泡图
```

具体步骤如下：

1）准备数据：这里暂以手动输入数据的方式设置向量 id 为品类名称，conver 为转化率，

pv为页面浏览量，price为客单价。

2）设置类目颜色：加载RColorBrewer包，并使用Spectral调色板设置10个样本点的颜色。

3）设置散点大小：设置散点大小参数cex的最大值cex.max和最小值cex.min。依据客单价数据price把各个散点大小（cex2向量）平均设置在[cex.max , cex.min]范围内。

4）绘制气泡图：设置plot函数的cex参数，并使用ylim设置y轴的范围。使用legend函数设置2行5列共10个品类的图例，并放于绘图的左上角。使用axis函数设置y轴的尺度标记为1%～5%（paste（seq（0,5,1）,"%",sep=""））。使用text函数在各个散点的中心绘制price数据：text（x=pv,y=conver,labels=price,cex=0.8），文字大小为0.8。最后在右上角绘制Z轴的含义：Z-客单价。

图形效果如图3-34所示。

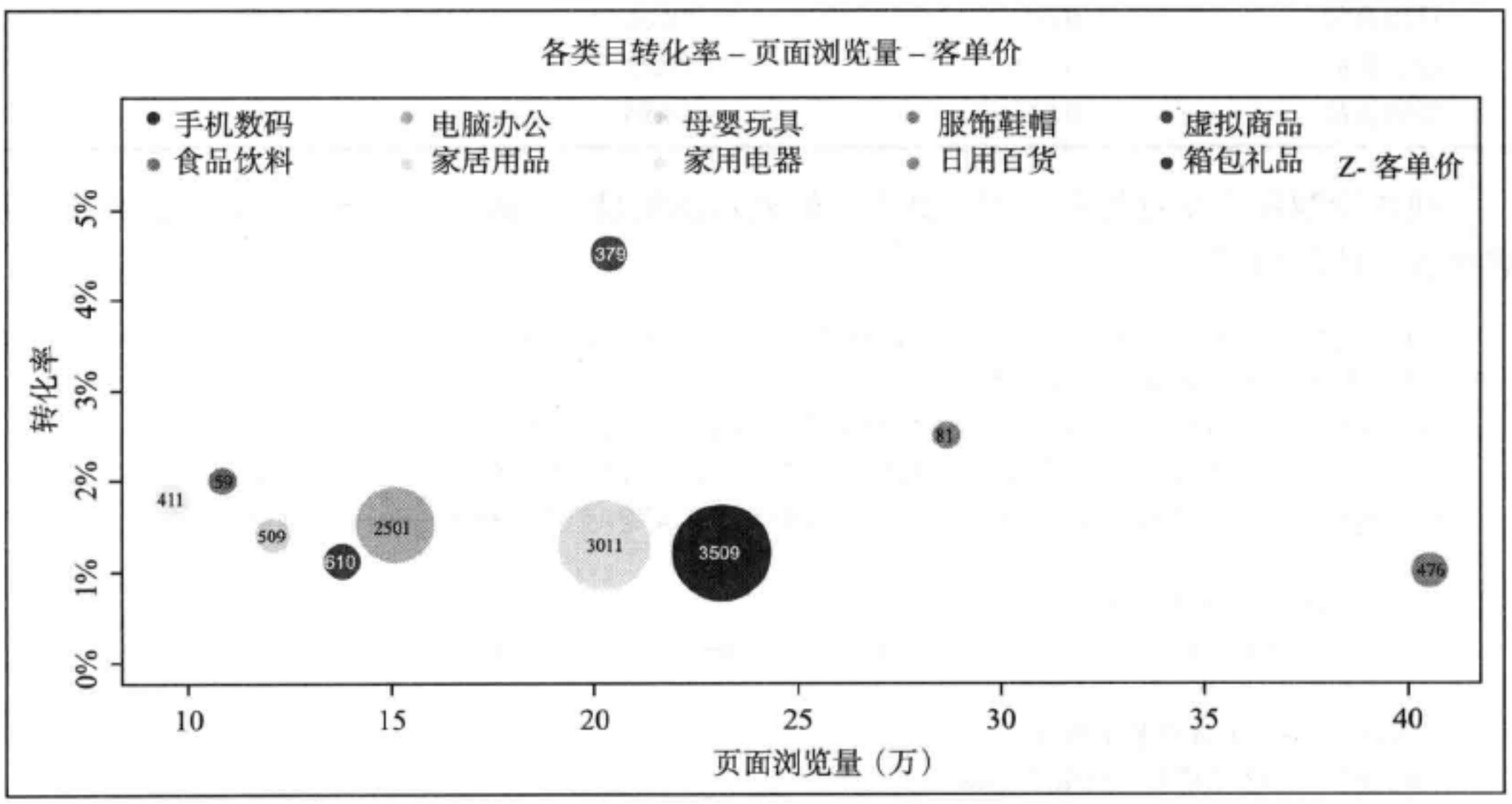


图3-34 气泡图实例

通过图3-34，可以直观地看到各个品类商品的基本销售差异：虚拟商品的转化率最高，服饰鞋帽的单品页浏览量最高，手机数码、电脑办公以及家用电器则有较高的客单价。这体现了不同类目商品本身的特质。例如，虚拟商品的信誉成本较低并且没有物流环节，这直接导致了该类商品的转化率比其他类商品高。网站可以依据不同特性，有针对性地运营不同类目商品，提高用户体验。

3.4.4 曲线图

1. 基本绘制方式

上面介绍了以点的方式展示样本数据（散点图和气泡图），接下来通过设置plot函数的

type 参数把样本的散点串联起来变成曲线。在绘制曲线图时，type 参数可以设置为如下值：

- 'l'：表示画线。
- 'b'：表示同时画点和线，但点线不相交。
- 'c'：表示将 type= 'b' 中的点去掉，只剩下相应的线条部分。
- 'o'：表示同时画点和线，且相互重叠（这是与 type = 'b' 的区别）。
- 's'：表示画阶梯线，从一点到下一点时，先画水平线，再画垂直线。
- 'S'：也是画阶梯线，但从一点到下一点是先画垂直线，再画水平线。

type 样式的具体内容可以参考 3.3.7 节“点”。图 3-35 是不同类型的线样式。

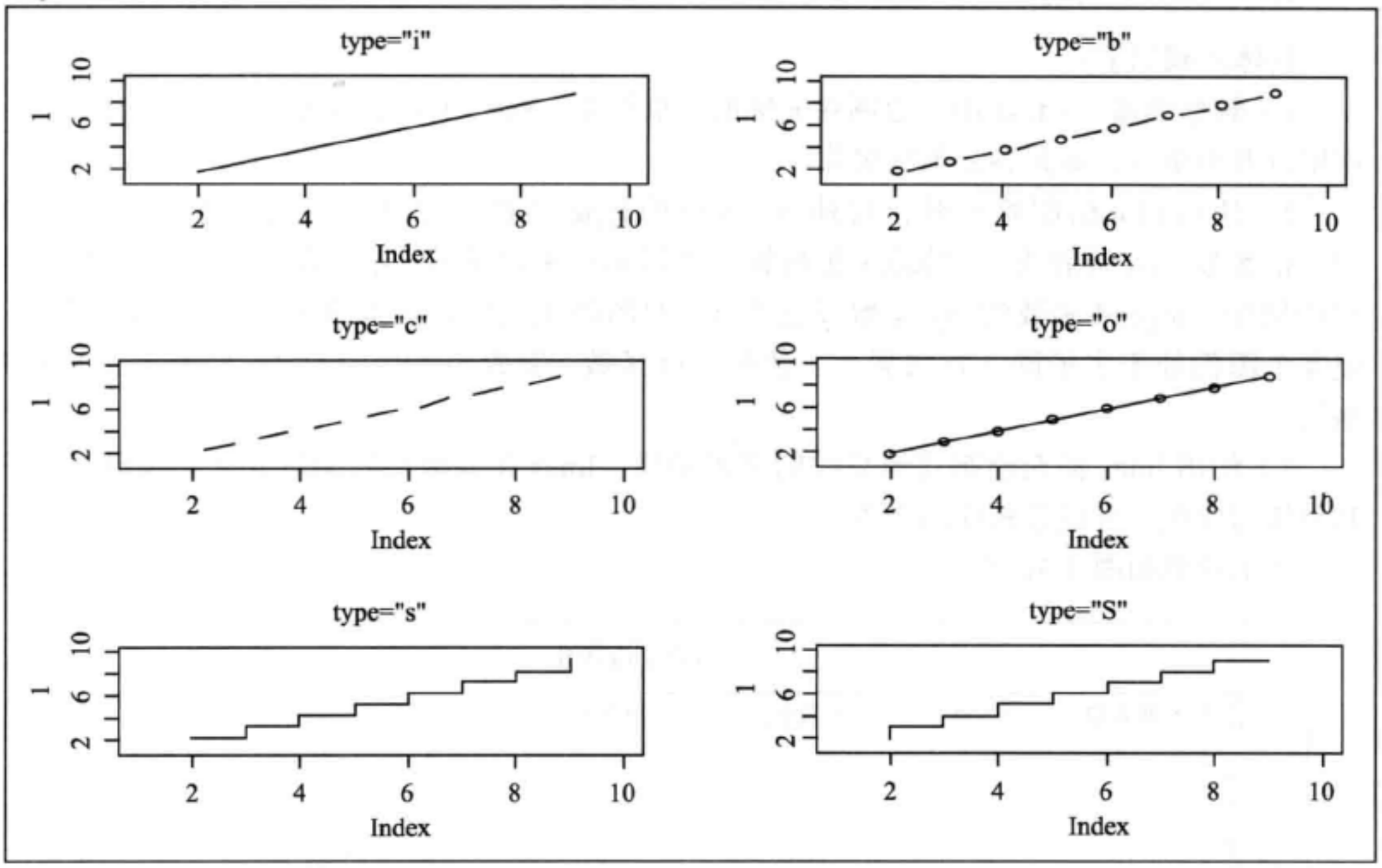


图 3-35 设置不同 type 参数值时的样式

除了使用 plot 函数外，绘制多条曲线时，也可以使用低级绘图函数 lines，其相关说明请参考 3.3.8 节“线”。

2. 实例：不同品类商品的月销量趋势图

以下是某电商网站 2012 年帆布鞋、T 恤、皮包、冲锋衣等 4 个品类的每月销量。

```
x.text<-c("1月","2月","3月","4月","5月","6月","7月","8月","9月","10月","11月","12月")
sales.1<-c(49.9,71.5,106.4,129.2,144.0,176.0,135.6,148.5,216.4,194.1,95.6,54.4)
sales.2<-c(83.6,78.8,98.5,93.4,106.0,84.5,105.0,104.3,91.2,83.5,106.6,92.3)
sales.3<-c(48.9,38.8,39.3,42.4,47.0,48.3,62.0,59.6,52.4,65.2,59.3,53.0)
sales.4<-c(42.4,33.2,34.5,39.7,52.6,70.5,57.4,62.0,47.6,39.1,46.8,51.1)
```

```
    id=c("帆布鞋","T恤","皮包","冲锋衣")  ——→①数据准备

    col=c("black","red","orange","purple")
    plot(sales.1,type="b",xaxt="n",ylim=c(0,300),col=col[1],main="月 销 量 趋 势 图",
xlab="月份",ylab="销量(万元)")
    axis(1,at=1:12,labels=x.text,tick=FALSE);
    legend("topleft",legend=id,horiz=T,pch=15,col=col,cex=0.8,bty="n")
    grid(nx=NA, ny=8,lwd=1,lty=2,col="blue")  ——→②使用plot绘制帆布鞋销售曲线

    lines(sales.2,type="b",col=col[2])
    lines(sales.3,type="b",col=col[3])
    lines(sales.4,type="b",col=col[4])  ——→③使用lines补充绘制其他品类的销售曲线
```

具体步骤如下：

1）数据准备：x.text用于绘制在x轴的尺度标识，sales.1-sales.4是4个品类的销售量，这里以万为单位，id是各品类的名称。

2）使用plot绘制帆布鞋销售曲线。plot的type参数设为"b"，使散点图串联成曲线图。把参数xaxt设置为n，隐藏x坐标轴，使用axis函数显示x轴并设置其坐标尺度标记。在图例中，legend函数的bty参数设置为n，即图例无边框，设置参数legend=id,horiz=T，使4个图例处于水平同一行来展示。使用grid函数（参数nx=NA, ny=8）绘制8条水平网格线。

3）使用lines补充绘制其他品类的销售曲线：lines在高级绘图函数plot的基础上补充其他销售曲线，并设置相应的颜色。

图形效果如图3-36所示。

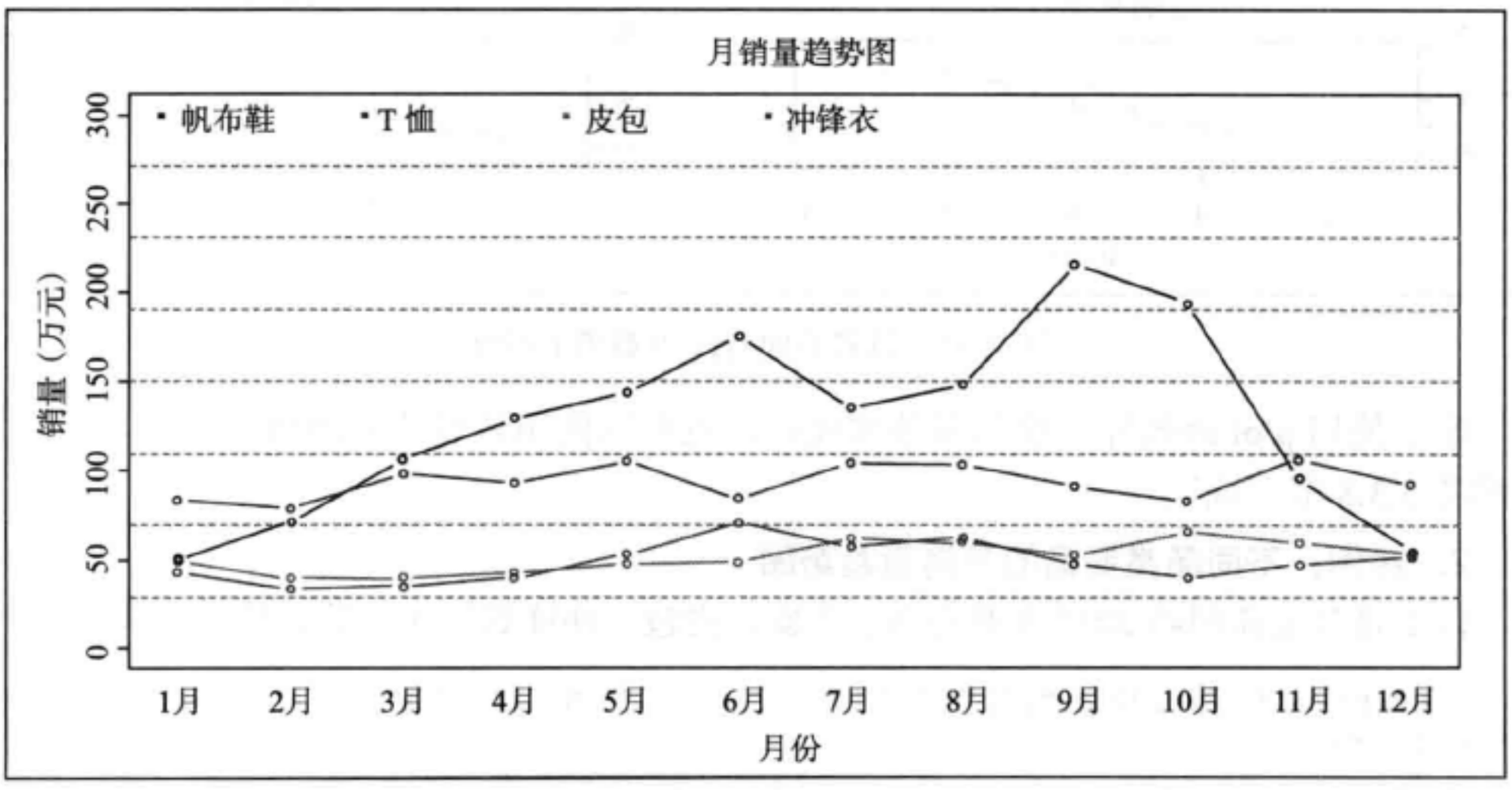


图3-36 月销量趋势图

曲线图可以清晰地展示数据的发展趋势，当样本点较少时，可以在图表上标记出各样本

的数值。当样本点较多时，不便于标记样本数值，这时可以通过 grid 函数绘制网格线来大致体现各样本点的数据大小。

3.4.5 柱状图

1. barplot 函数

柱状图的实际还是散点图，只是它更适合用于描述主数据（y 轴数据）在不同分类（x 轴是因子类型的分类数据，或者是时间数据）下的展示情况。同时，柱状图还可以清晰地绘制不同情况下的多组数据，以便于横向对比。barplot 函数是柱状图的核心函数，前面在 3.3.1 节的引例中已经简单讲解过。其基本形式是：barplot（height, horiz = FALSE，beside = FALSE,…），参数如表 3-15 所示。

表 3-15 barplot 函数的参数

参数	说明
height	绘图的数据 如果绘制一组数据，则以向量形式输入 如果绘制对组数据，则以矩阵形式输入，每行表示一组数据
horiz	如果是 FALSE（默认取值），则绘制柱状图 如果是 TRUE，则绘制条形图
beside	如果是 FALSE（默认取值），则不同组数据垂直堆积展示 如果是 TRUE，则不同组数据水平并列展示

下面再次回顾一下 3.3.1 的引例，以详细说明以上 3 个参数的含义。

数据：date	data$pre（上周 pv 数据，单位：万）	data$now（本周 pv 数据，单位：万）
周一	113	129
周二	134	122
周三	123	134
周四	145	149
周五	137	146
周六	196	215
周日	187	208

（1）height 参数

如果只想绘制本周的 pv 数据，则把参数 height 设置为向量 data$now。

```
par(mfrow=c(2,1))
data<-data.frame(pre=c(113,134,123,145,137,196,187),now=c(129,122,134,149,146,
215,208));
col=c("azure4","brown4")
barplot(data$now,beside=TRUE,col=col[2],axes=F)
axis(2)
text.x<-c("周一","周二","周三","周四","周五","周六","周日")
axis(1, seq(from=0.7,by=1.2,length.out=7), labels = text.x, tick = FALSE, cex.
```

height= 向量 data$now

```
axis = 0.75)
    title(main=list("以向量形式输入一组数据",cex=1.5,col="red",font=3),
          ylab="网站日页面浏览量pv")
```

如果想绘制上周和本周的两组 pv 数据，则在 height 参数中输入矩阵，其行数据分别是上周 pv（data$pre）和本周 pv（data$now），使用 t（as.matrix（data））可以产生符合要求的矩阵，如下所示：

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]
pre	113	134	123	145	137	196	187
now	129	122	134	149	146	215	208

代码如下：

```
barplot( t(as.matrix(data)),beside=TRUE,col=col,axes=F)
axis(2)                                          height=矩阵 t(as.matrix(data))
text.x<-c("周一","周二","周三","周四","周五","周六","周日")
axis(1, c(2,5,8,11,14,17,20), labels = text.x, tick = FALSE, cex.axis = 0.75)
title(main=list("以矩阵形式输入多组数据",cex=1.5,col="red",font=3),
      ylab="网站日页面浏览量pv")
```

图形效果如图 3-37 所示。

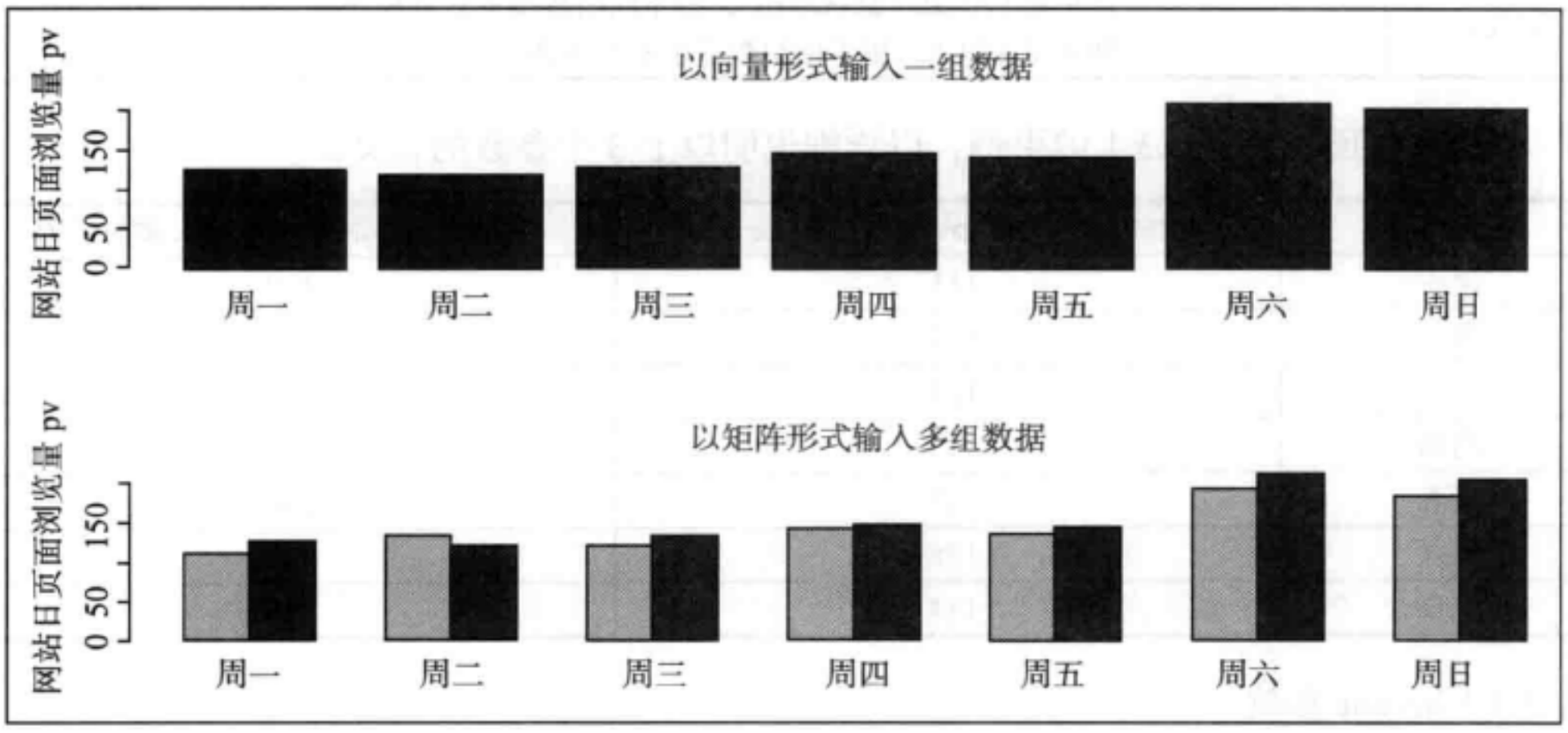


图 3-37 结果展示

（2）horiz 参数

上面介绍的各图均是垂直于 x 轴的柱状图，如果设置 horiz 为 TRUE，则绘制平行于 x 轴的条形图。代码如下：

```
barplot( t(as.matrix(data)),beside=TRUE,col=col,axes=F,horiz=T)
axis(1)                                                  设置绘制条形图
text.x<-c("周一","周二","周三","周四","周五","周六","周日")
```

```
axis(2, c(2,5,8,11,14,17,20), labels = text.x, tick = FALSE, cex.axis = 0.75)
title(main=list("horiz=TRUE 时绘制条形图 ",cex=1.5,col="red",font=3),
      ylab=" 网站日页面浏览量 pv")
```

图形效果如图 3-38 所示。

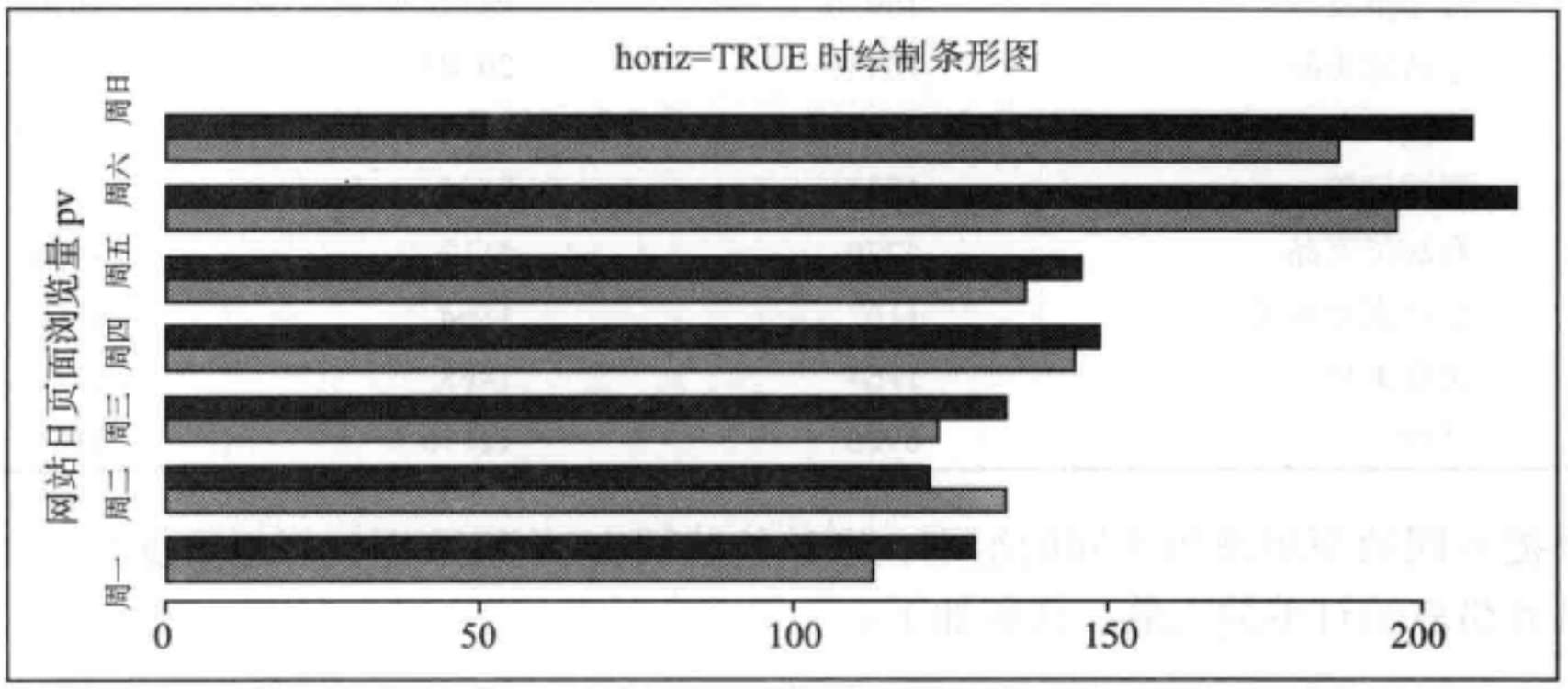


图 3-38 结果展示

（3）beside 参数

当有多组数据展示时，可通过 beside 参数把各组数据以水平并列的方式（为 FALSE，默认取值）展示或以垂直堆积方式（为 TRUE）展示，如图 3-39 所示。

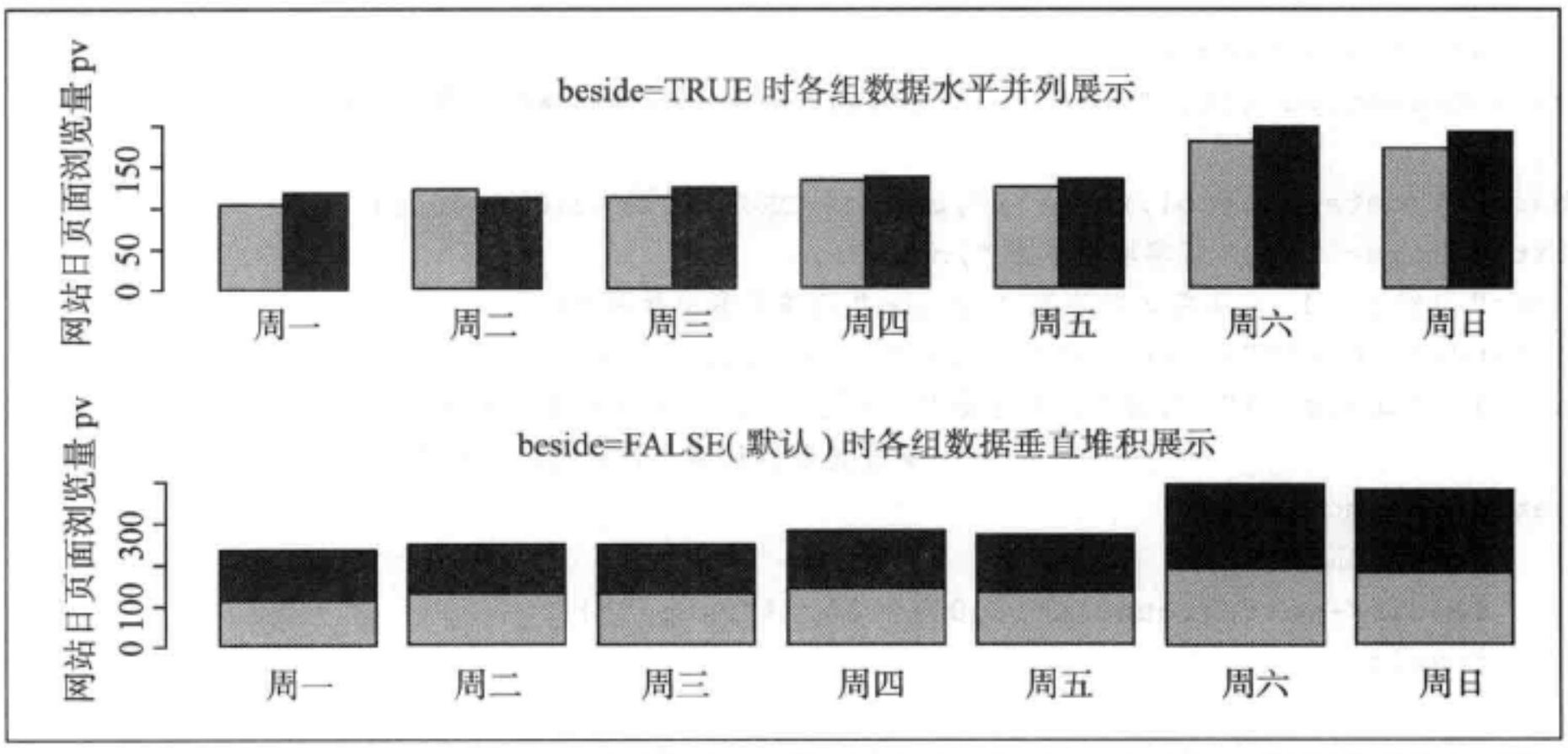


图 3-39 结果展示

2. 实例：展示订单取消原因 – 每月数据

以下是某网站研究用户取消订单原因时的统计数据。

数据：	原因	3月份数据	4月份数据	5月份数据
	订购错误	25746	46496	53356
	支付失败	8595	20150	26547
	更换其他商品	12832	19682	23271
	订单重复	10910	14177	16909
	忘选优惠品	7043	20703	14789
	等待太久	2978	8434	12733
	测试订单	6934	9560	11545
	有缺货商品	4770	5113	7483
	已在别处购买	1137	1804	2506
	价格太贵	1164	1468	1743
	其他	6926	11156	11869

如果把不同的原因视为不同的组别，则该柱状图有11组数据，每组数据代表不同原因下3～5月份取消订单的人数。代码如下：

```
    id<-c("订购错误","支付失败","更换其他商品","订单重复","忘选优惠品","等待太久","测试
订单","有缺货商品","已在别处购买","价格太贵","其他")
    month.3<-c(25746,8595,12832,10910,7043,2978,6934,4770,1137,1164,6926)
    month.4<-c(46496,20150,19682,14177,20703,8434,9560,5113,1804,1468,11156)
    month.5<-c(53356,26547,23271,16909,14789,12733,11545,7483,2506,1743,11869)
    data<-matrix(c(month.3,month.4,month.5),ncol=3) ——————→①数据准备

    library(RColorBrewer)
    col<-brewer.pal(11, "Spectral")[1:11] ——————————→②设置不同组数据的颜色

    barplot(data,col=col,xaxt="n",beside=TRUE,ylim=c(0,60000))
    title(main=list("订单取消原因",cex=2),
    sub="月份:3-4    品类：帆布鞋",ylab="订单月取消数目")
    legend("topleft",legend=id,pch=15,col=col,ncol=2,cex=0.8)
    axis(1,labels=c("3月份","4月份","5月份"),at=c(6,18,30),tick=FALSE)
                                    ——————→③绘制柱状图，设置标题、图例、坐标轴
    per100<-function(x){
        x<-x/sum(x)
        result<-paste(round(x*10000)/100,"%",sep="")
        result
    }
    text(labels=c(per100(month.3),per100(month.4),per100(month.5)),cex=0.7,
    x=c(seq(from=1.5,by=1,length.out=11),seq(from=13.5,by=1,length.out=11),seq(from
=25.5,by=1,length.out=11)),
        y=c(month.3,month.4,month.5)+1000)
                                    ——————→④在柱状图内绘制取消人数的百分比数据
```

具体步骤如下：

1）数据准备：由于柱状图的核心函数 barplot 在展示多组数据时使用矩阵对象作为其源数据的输入，所以这里建立矩阵对象 data，其第 1 ～ 11 行分别表示每种订单取消原因 3 ～ 5 月份的用户数。

2）设置不同组数据的颜色：使用 RColorBrewer 包的 Spectral 调色板内的 11 种颜色。

3）绘制柱状图，设置标题、图例、坐标轴：使用 barplot 绘制柱状图，并设置参数 beside=TRUE，把各组柱状图数据通过水平排列方式展示。

图形效果如图 3-40 所示。

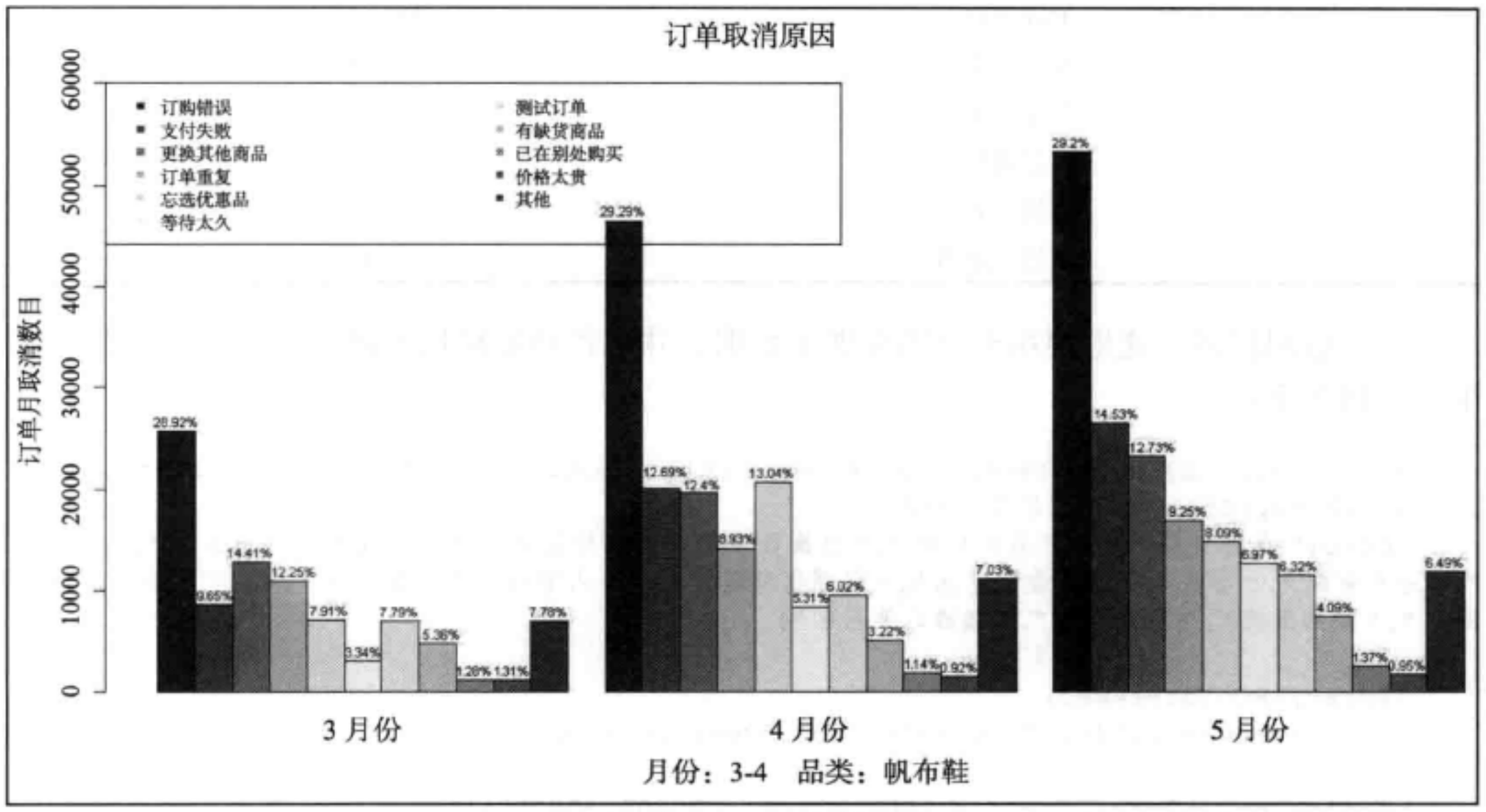


图 3-40 订单取消原因柱状图

3.4.6 条形图

在 3.4.5 节柱状图的基础上，把 barplot 函数的 horiz 参数设置为 TRUE 便可绘制条形图。

实例：个人中心各模块用户点击率

以下是某网站个人中心内各个功能模块 9 月份的页面浏览量。

数据：	功能模块	9 月份页面浏览量
	我的个人中心	29123
	我的订单	279750
	退换货办理	89994
	海外订单	15851
	我的收藏	61741
	礼品卡	119477

（续）

数据：	功能模块	9月份页面浏览量
	优惠券	75301
	账户余额	65602
	会员积分	75924
	收货地址	37943
	促销信息退订	11952
	个人资料	34567
	关联账户	11894
	账户安全	42780
	商品评论	18511
	商品提问	9450
	邀请方式	1011
	邀请记录查询	1533

由于组别较多，这里使用条形图来展示数据，并把各功能模块流量的百分比展示在图表中。代码如下：

```
    pv<-c(29123,279750,89994,15851,61741,119477,75301,65602,75924,37943,11952,34567
,11894,42780,18511,9450,1011,1533)
    id<-c("我的个人中心","我的订单","退换货办理","海外订单","我的收藏","礼品卡","优惠券
","账户余额","会员积分","收货地址","促销信息退订","个人资料","关联账户","账户安全","商品
评论","商品提问","邀请方式","邀请记录查询")

    library(RColorBrewer)
    col<-c(brewer.pal(9, "YlOrRd")[1:9],brewer.pal(9, "Blues")[1:9])

    barplot(pv,col=col,horiz = TRUE,xlim=c(-30000,300000))
    title(main=list("个人中心各模块用户点击率",cex=2),sub="时间:2012.9.1-2012.9.30",
ylab="个人中心各模块")
    text(y=seq(from=0.7,length.out=18,by=1.2),x=-15000,labels=id)
    legend("topright",legend=rev(id),pch=15,col=rev(col),ncol=2)

    text(labels= paste(round(10000*pv/sum(pv))/100,"%",sep=""),cex=0.7,
         y=seq(from=0.7,length.out=18,by=1.2),
         x=pv+10000)
```

在上面的例子中，y轴的尺度标记并未使用axis函数设置，原因在于使用axis函数设置的标记其文字方向是垂直的，这会导致y轴没有充足的空间展示尺度标记。而使用text函数设置的标记文字是水平方式排列的，它并不会占据太多y轴的空间。

paste（round（10000*pv/sum（pv））/100,"%",sep=""）命令会把pv数据转化为小数点后2位的百分比字符串，把pv向量c（29123,279750,89994,…）转化为字符串向量c（"2.96%","28.48%","9.16%",…）。

图形效果如图3-41所示。

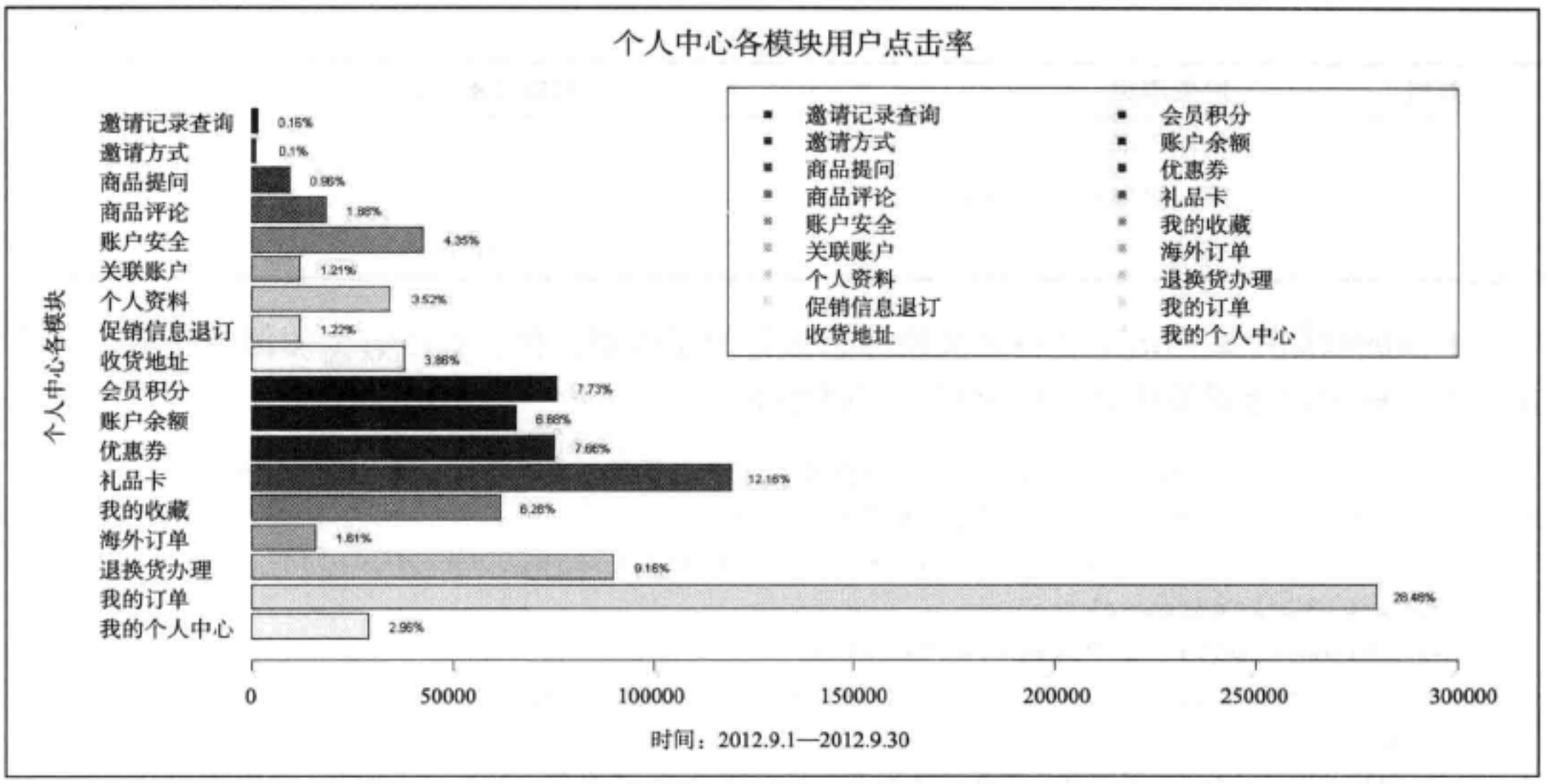


图 3-41 各模块用户点击率的条形图

由图 3-41 可知，用户在使用个人中心系统时主要使用完成购买流程的“我的订单”功能，其次是有关用户虚拟支付资源的礼品卡、优惠券，以及会员积分功能。可见个人中心承载的最核心功能仍然是帮助用户完成购物，其占比可以累积达到 50% 以上。退换货的 pv 占了 9.16%，可以推测用户在日常登录个人中心后台后的退换货行为比例。

3.4.7 饼图

饼图可以静态展示整体内不同成分的构成。实际上饼图也可以使用条形图或柱状图代替，并无本质区别，只是当成分数目较少时，使用饼图绘制会更好。当成分数目较多时，可以考虑使用柱状图和条形图展示。

饼图的核心函数是 pie，其基本形式为：

pie（x, labels = names（x）, ...）

在饼图中实际并没有散点图的 x 轴和 y 轴概念，这里的 x 参数是由各成分数据组成的向量，其数值用饼图的面积占比来表示。labels 用于定义各个成分的名称。

1. 实例：退货率原因分析

以下是某网站调研用户退货原因的统计数据。

数据：	退货原因	退货数量占比
	安全性得不到保障	0.053
	产品质量、售后服务	0.460
	付款不便	0.087
	送货耗时、渠道不畅	0.213

（续）

数据：	退货原因	退货数量占比
	价格不够诱人	0.078
	网上提供消息不可靠	0.042
	其他	0.067

上面的数据中共列出了7种退货原因数据，为了观察方便，这里省去图例的展示，直接在饼图上标识出退货原因和占比情况。代码如下：

```
data<-data.frame(id=c("安全性得不到保障","产品质量、售后服务","付款不便","送货耗时、渠道不畅","价格不够诱人","网上提供消息不可靠","其他"),
                 num=c(0.053,0.46,0.087,0.213,0.078,0.042,0.067))
library(RColorBrewer)
col<-brewer.pal(11, "Spectral")[3:11]
pie(data$num,col=col,xaxt="n",labels=paste(data$id,":",round(data$num*10000)/100,"%",sep=""))
title(main=list("网站退货率高的原因",cex=2),sub="月份:3-4    品类：帆布鞋")
```

在pie函数中设置xaxt参数为"n"，隐藏x轴的展示。设置labels为paste（data$id,":",round（data$num*10000）/100,"%",sep=""），直接在饼图各个部分绘制退换货原因和数据占比。未设置参数名称时，pie函数的第一个参数表示x，即饼图各成分的数据，这里是源数据data的num列向量（使用data$num表示）。需要注意的是，由于pie函数绘制的图形没有x轴和y轴的概念，所以尽量不要与plot/barplot/lines/text等需要设置x轴和y轴数据的函数混合使用。图形效果如图3-42所示。

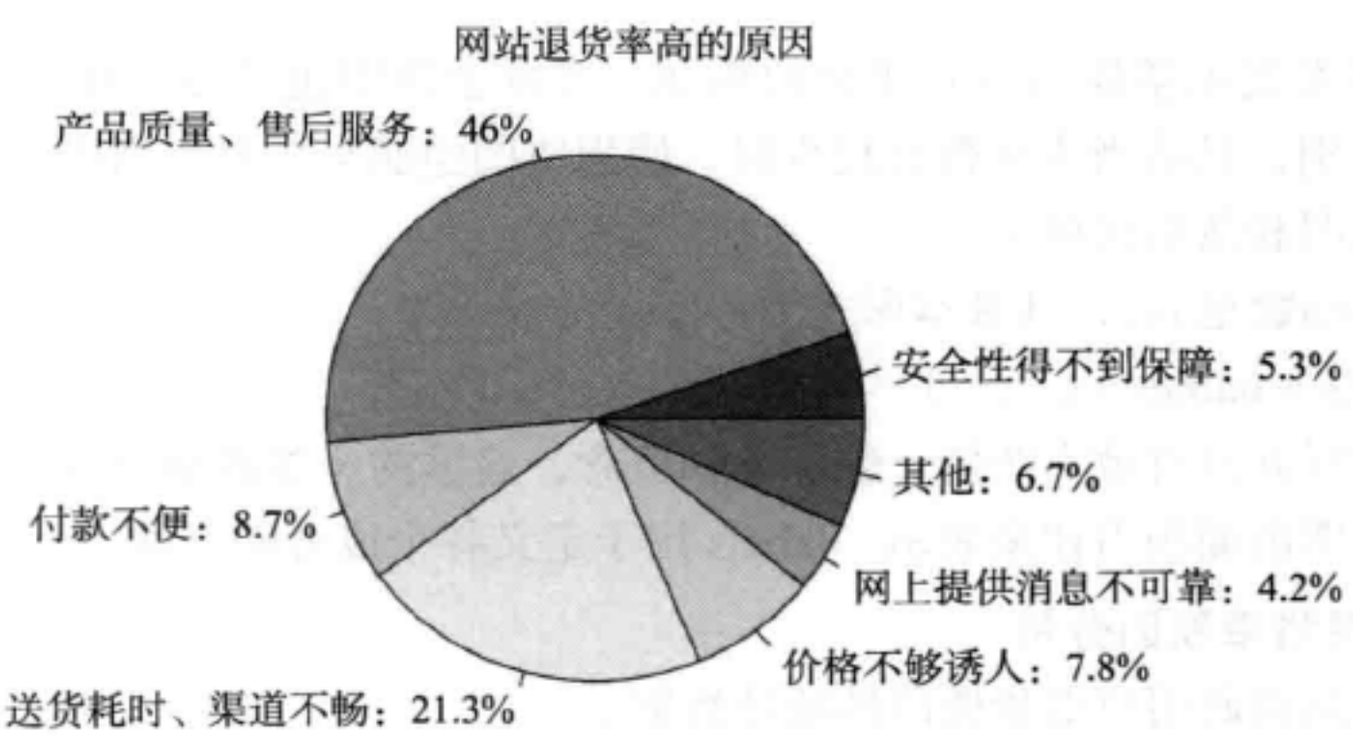


图3-42 退货原因饼图

观察饼图中各个部分的占比可知，大部分用户的退货原因是产品质量和售后问题（占46%），其次是物流问题（占21.3%）。值得关注的是，安全性占5.3%以及网上信息不可靠占

4.2%，这表明用户在购物中对网站的信誉成本也很敏感，需要继续增强网站的品牌效应并做好口碑营销。

2. 复合图展示

当数据中的成分数较多，或者各成分数据的比例不均导致部分成分没有足够空间展示时，可以使用复合柱状图。

这里仍然使用上面的例子，并把其他退货原因选项细化为：物流服务人员态度不好、尺码不合适、发现更好的商品。并把上例中提到的原因数据作为主柱状图绘制于左侧，新细分的 3 个原因数据作为副柱状图绘制于右侧，如图 3-43 所示。

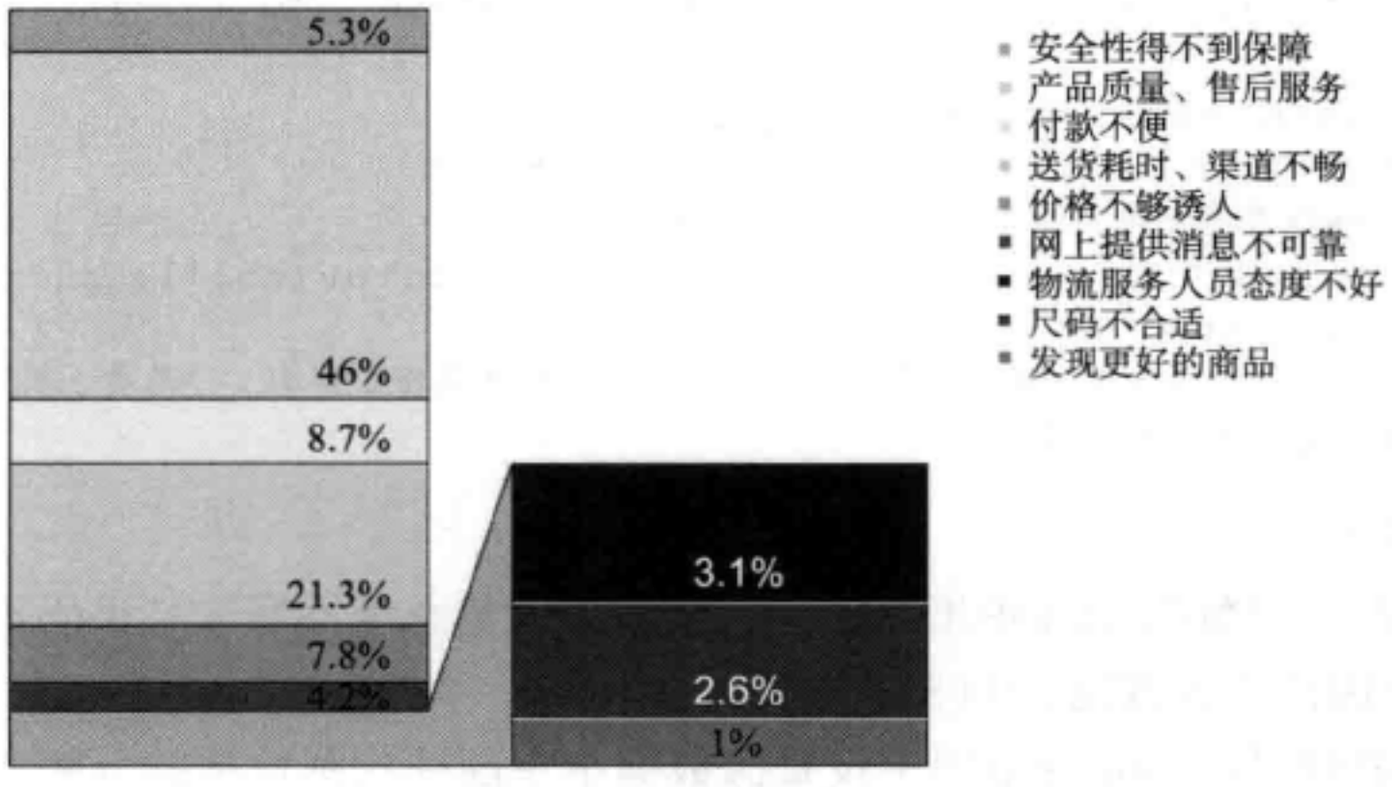


图 3-43 退货原因复合图

具体代码如下：

```
    data<-data.frame(id=c("安全性得不到保障","产品质量、售后服务","付款不便",
                          "送货耗时、渠道不畅","价格不够诱人","网上提供消息不可靠",
                          "物流服务人员态度不好","尺码不合适","发现更好的商品"),
                     num=c(0.053,0.46,0.087,0.213,0.078,0.042,0.031,0.026,0.010))
```

①数据准备

```
    split<-6
    max.bar2<-0.4
```

②设置分割变量和副柱状图高度变量

```
    bar1.data<-matrix(rev(c(rep(NA,split+1),data$num[1:split],sum(data$num[-
(1:split)]))),ncol=2,nrow=split+1)
    bar2.data<-matrix(c(rep(NA,split+1),rev(data$num[-(1:split)]),rep(NA,
nrow(data)-split+1)),ncol=2,nrow=split+1)
```

③转化主副柱状图数据

```
    library(RColorBrewer)
```

```
    col<-brewer.pal(11, "Spectral")[3:11]
    barplot(bar1.data,col=c("azure3",col[1:split]),axes=FALSE,ylim=c(0,1),xlim=c(0,
4.5),border="azure3") ——→绘制主柱状图
    barplot(bar2.data*(max.bar2/sum(data$num[-(1:split)])),col=col[-(1:split)],axes
=FALSE,add=TRUE,border="azure3") ——→绘制副柱状图
    polygon(x=c(1.2,1.2,1.4,1.4),y=c(0,sum(data$num[-(1:split)]),max.bar2,0),col="a
zure3",border="azure3") ——→绘制连接主副柱状图关系的多边形
                    └→④绘制复合柱状图主体
    labels=paste(round(data$num*10000)/100,"%",sep="")
    y1<-0 ——→主柱状图绘制百分比数据的y数值
    for(i in 1:split){y1[i]=sum(data$num[-(1:i)])}
    text(x=1,y=y1+0.02,labels[1:split],cex=0.8)
    y2<-0 ——→复柱状图绘制百分比数据的y数值
    for(i in 1:(nrow(data)-split-1)){y2[i]=sum(data$num[(split+i+1):nrow(data)])}
    y2<-c(y2,0)
    y2<-y2*(max.bar2/sum(data$num[-(1:split)]))
    text(x=2,y=y2+0.02,labels[-(1:split)],cex=0.8)
            └→⑤在图形中绘制各成分的百分比数据
    legend("topright",legend=data$id,pch=15,col=c(rev(col[1:split]),rev(col[-
(1:split)])),ncol=1,bty="n")
    title(main=list("网站退货率高的原因",cex=2),sub="月份:3-4     品类:帆布鞋")
          └→⑥绘制图例和标题
```

具体步骤如下：

1）数据准备：源数据 data 中增加了“物流服务人员态度不好”、“尺码不合适”、“发现更好的商品”原因的占比数据：0.031、0.026、0.010。

2）设置主要变量：split 变量用于设置源数据中主副柱状图数据的分割点，本例中设置为 6，data$num[1:6] 以及 sum（data$num[7:9]）为主柱状图数据，data$num[7:9] 为复合柱状图。max.bar2 变量为绘制的副柱状图的最高高度。实际可以把 split 和 max.bar2 变量设置为参数，并把后续的代码封装成绘制复合柱状图的函数，有兴趣的读者可以自己尝试一下。

3）转化主副柱状图数据：依据分割变量 split，把源数据 data 转化为可用于绘制主柱状图（bar1.data）和副柱状图（bar2.data）的数据，如图 3-44 所示。

```
> bar1.data
      [,1] [,2]
[1,] 0.067   NA
[2,] 0.042   NA
[3,] 0.078   NA
[4,] 0.213   NA
[5,] 0.087   NA
[6,] 0.460   NA
[7,] 0.053   NA
```

```
> bar2.data
      [,1]  [,2]
[1,]   NA 0.010
[2,]   NA 0.026
[3,]   NA 0.031
[4,]   NA    NA
[5,]   NA    NA
[6,]   NA    NA
[7,]   NA    NA
```

图 3-44 源数据 data 转化后

NA 部分的数据实际是不会在窗口中绘制出来的。例如，使用 bar2.data 数据绘制副柱状图 barplot 时，由于第一列数据均是 NA，所以 x=1 处（左侧）不会绘制任何图形。这保证绘制副柱状图时不会覆盖主柱状图。对于第二列，也只绘制 bar2.data[1:3,2] 3 个数据。

4）绘制复合柱状图主体：准备好数据后，使用两个 barplot 函数分别绘制主副柱状图，其中绘制副柱状图的 barplot 函数设置参数 add 为 TRUE，表明该次操作是在之前绘图的基础上完成的。把被细化的成分放于主柱状图的最底侧并使用 polygon 函数绘制一个梯形多边形，用以标示该部分和副柱状图的细分关系。

5）在图形中绘制各成分的百分比数据：labels 是绘制的文字，它把各成分数据转化为百分比数据。y1 和 y2 是绘制数据时的 y 轴位置，其数值是各个成分所在矩形底部的 y 轴数值加偏量 0.02。

6）绘制图例和标题：可以参考前面各章节的实例，这里就不再详述了。

3.4.8 面积堆积图

在很多情况下，我们希望动态了解整体在不同阶段下的不同成分构成。当不同阶段的水平值较少（数据分组的数目较少）时，可以使用条形图或柱状图。但如果数目较多，使用条形图和柱状图将会很拥挤，甚至会出现在一张图中放不下的情况，此时可以使用面积堆积图来表示。

绘制面积堆积图需要加载 ggplot2 包，ggplot2 包绘制图形的语法相对统一，它主要是弥补了 R 语言内的自带绘图函数语法相对零散的缺点。用户可以使用该包绘制各种新颖的图形。

1. 基本绘制方式

加载了 ggplot2 包后，可以：

- 绘制百分比面积堆积图：ggplot(data=… ,mapping=aes(…))+ geom_area(position="fill")
- 绘制面积堆积图：ggplot(data=… ,mapping=aes(…))+ geom_area()

ggplot 函数的参数设置如表 3-16 所示。

表 3-16 ggplot 函数的参数

参数		说明
data		设置面积堆积图的源数据，通常为数据框
mapping		设置源数据中与 x 轴、y 轴以及分组和分组颜色的关系。通常使用：aes（x=…,y=…,group=…,fill=…）来取值
aes 内部参数	x	指定源数据 data 中被映射为 x 轴的列
	y	指定源数据 data 中被映射为 y 轴的列
	group	指定源数据 data 中用于区分不同组别数据所在的列
	fill	指定源数据 data 中用于区分不同组别颜色数据所在的列

ggplot 函数中的 data 参数的格式，建议使用如下形式：

```
data = data.frame(x= ,y= ,group=)
```

并把 mapping 参数设置为：

```
mapping = aes(x=x,y=y,group=group,fill=group)
```

这里先使用柱状图近似地研究面积堆积图，下面是一组用于设置 data 参数的数据框对象实例。在绘制好面积堆积图后，如果将其放大，只观察前三个柱状图，则数据和图形的对应关系如图 3-45 所示。

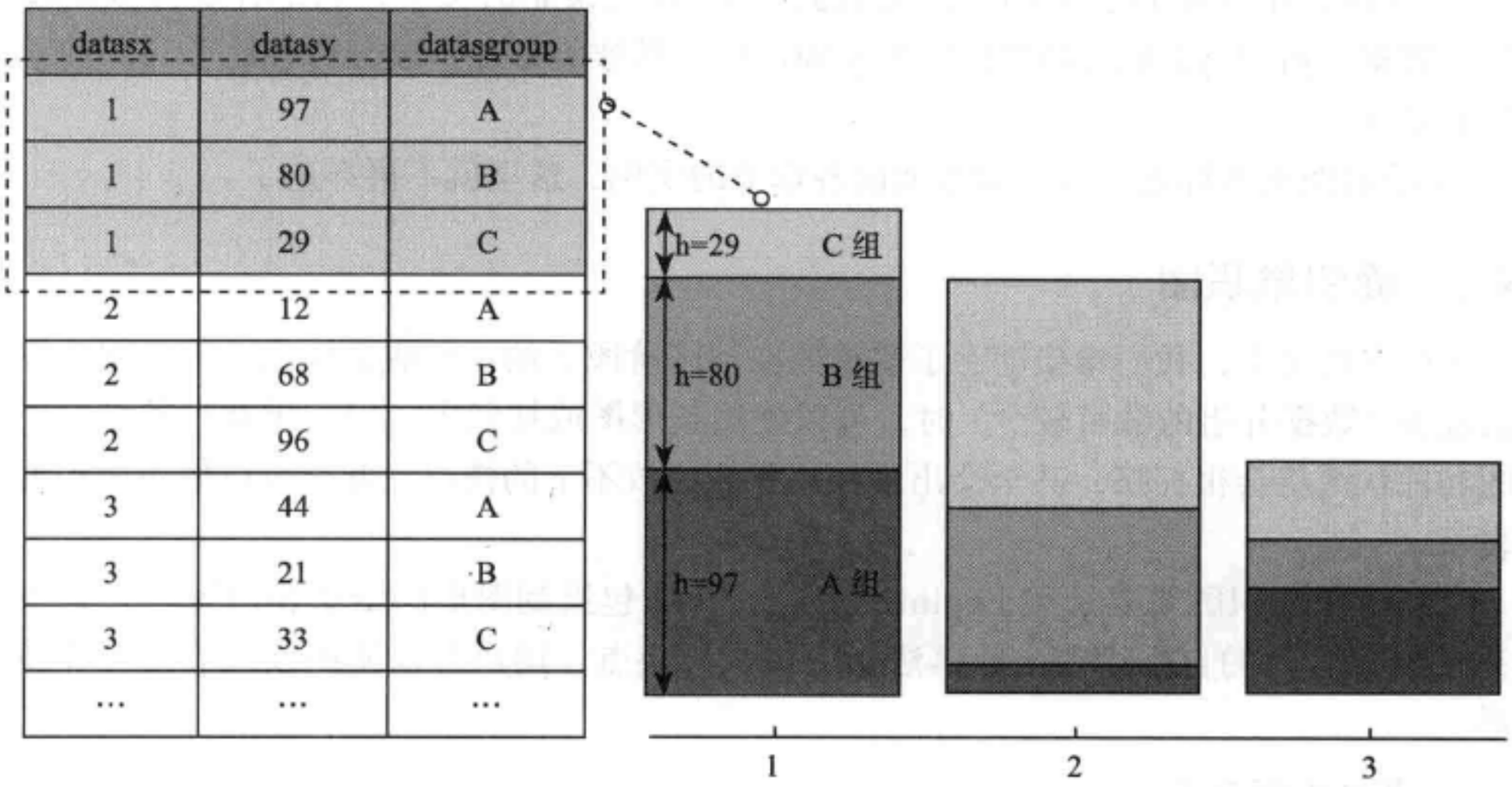


图 3-45 数据和图形的对应关系

2. 实例：展示订单取消原因 – 每日数据

在 3.4.5 的实例中，使用订单取消原因的每月数据来展示 3 ～ 5 月的月成分比例，这里要展示 3 ～ 5 月的每日成分变化，如图 3-46 所示。

	A	B	C	D	E	F	G	H	I	J	K
1	测试订单	等待太久	订单重复	订购错误	更换其他产	价格太贵	忘选优惠品	已在别处购	有缺货商品	支付失败	其他
2	311	43	251	571	336	29	154	22	57	298	184
3	341	22	210	541	263	13	144	22	49	224	137
4	234	42	215	527	263	27	127	24	50	220	116
5	193	44	153	494	261	14	153	30	38	215	137
6	109	29	155	443	208	19	120	26	36	248	98
7	100	39	189	487	191	20	95	23	36	276	94
8	228	29	170	545	250	13	190	21	52	231	142
9	173	33	192	536	228	16	146	25	60	230	142
10	293	30	184	525	292	19	152	32	36	208	152
11	259	29	172	525	255	21	176	17	50	238	162
12	241	47	226	526	276	23	157	27	34	305	163
13	120	27	260	575	256	26	159	28	222	234	141
14	130	20	462	631	296	18	158	30	50	356	164

图 3-46 订单取消原因分析（3.1—5.31）.csv 文件

在图 3-46 所示的文件中，每列代表某订单取消原因下 3 ～ 5 月份内每日的取消数量。使用如下代码来绘制堆积面积图。

```
root="C:/Users/liming/Desktop/写书/图形使用/"
file=paste(root,"订单取消原因分析(3.1-5.31).csv",sep="")
data.csv<-read.csv(file)
library(ggplot2)
t<-0;group<-"";num<-0
for(i in 1:nrow(data.csv)){
    t<-c(t,rep(i,ncol(data.csv)))
    group<-c(group,names(data.csv))
    num<-c(num,as.matrix(data.csv)[i,])
}
data<-data.frame(x=t,group=group,y=num)
data<-data[-1,]
ggplot(data, aes(x=x,y=y,group=group,fill=group)) + geom_area()
```

图形效果如图 3-47 所示。

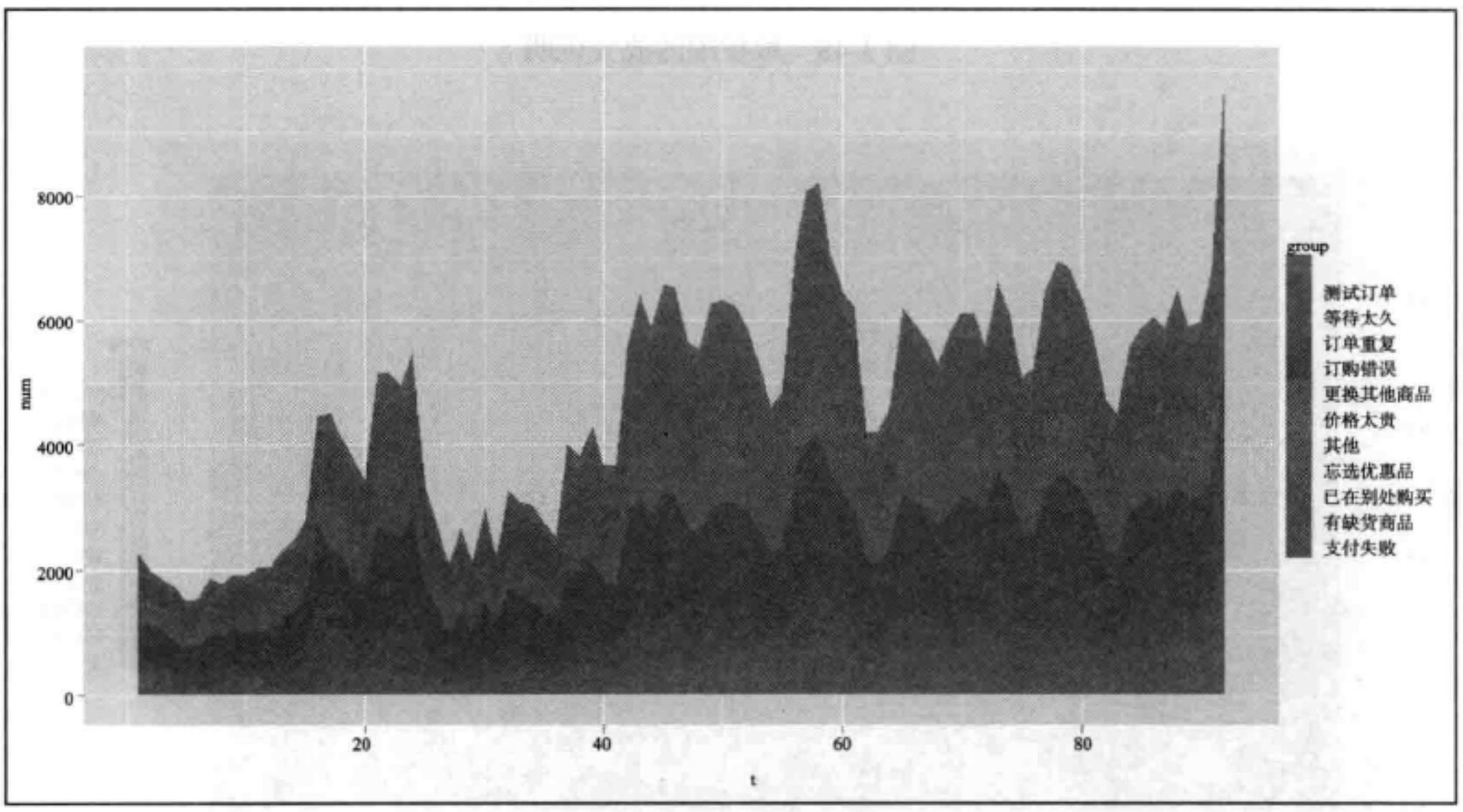


图 3-47 取消订单原因的面积堆积图

下面把左侧面积堆积图中的一小段进行放大，得到近似于柱状图的右侧图形，并标记出各个绘图元素与源数据 data 每列的关系，如图 3-48 所示。

如果把上面代码的最后一行替换为如下代码，则可以绘制百分比面积堆积图。

```
ggplot(data, aes(x=x,y=y,group=group,fill=group)) + geom_area(position="fill")
```

图形效果如图 3-49 所示。

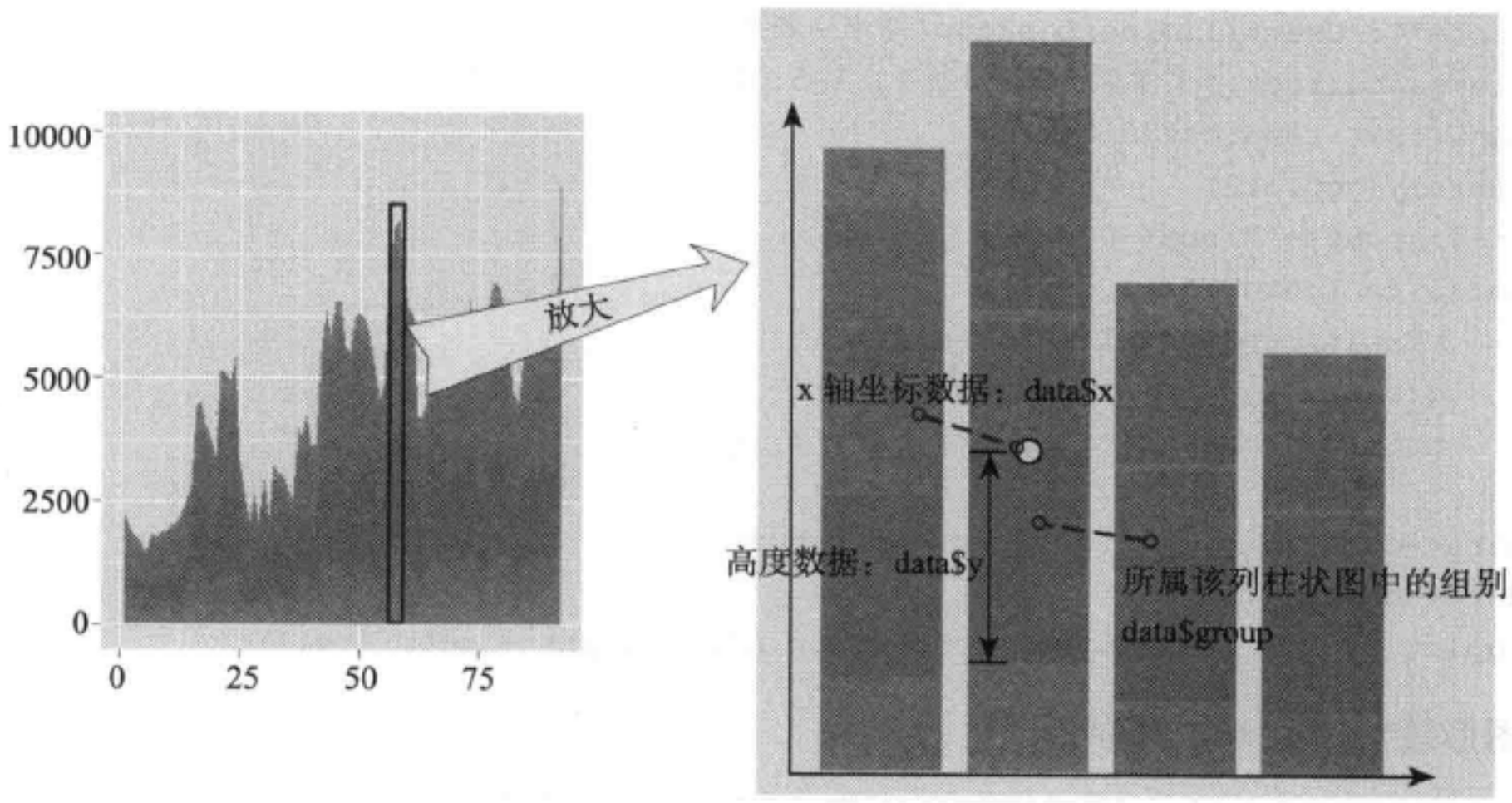


图 3-48 堆积图的放大说明

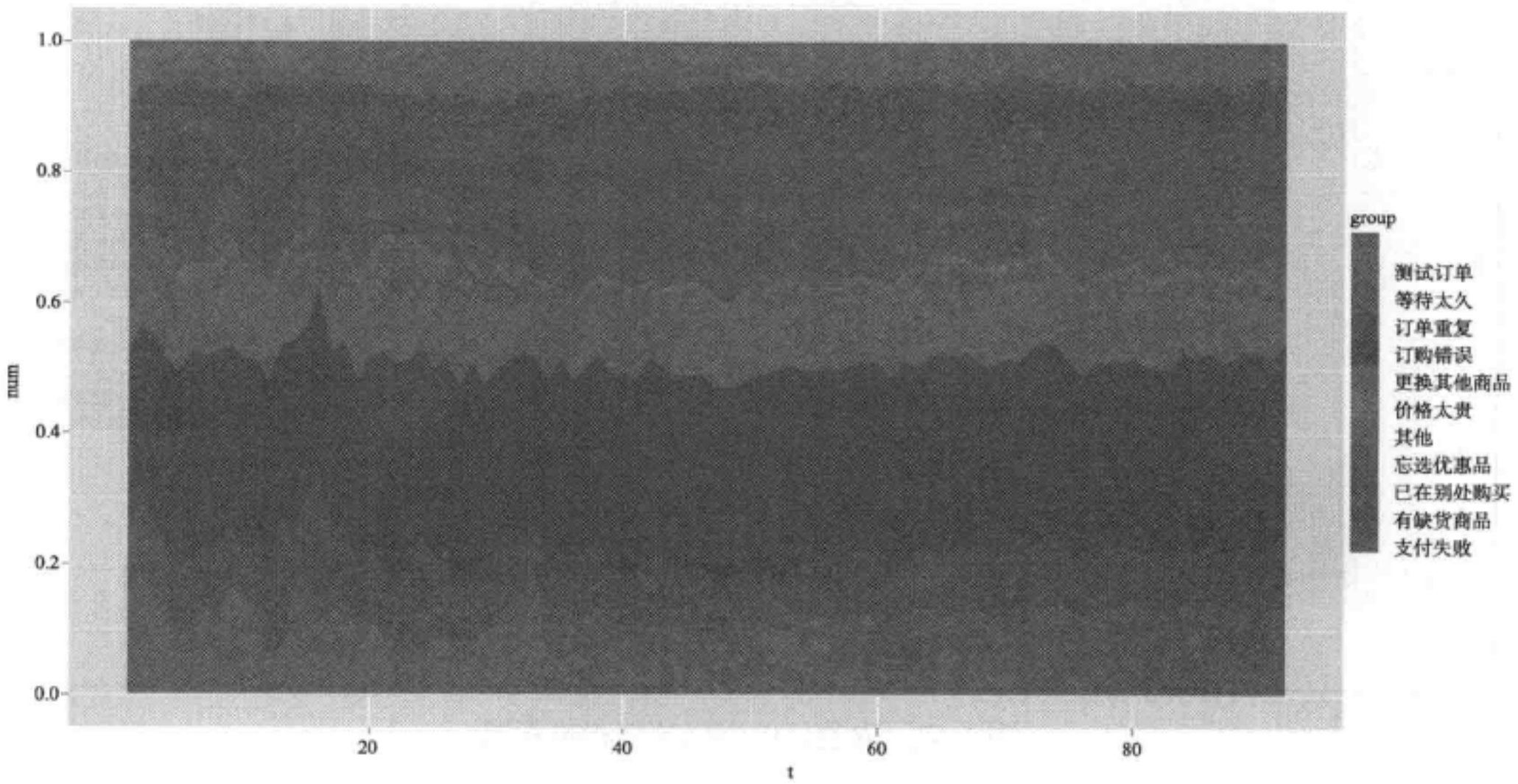


图 3-49 取消订单原因的面积堆积图

3.4.9 直方图和密度曲线图

如果想要研究某一变量的分布情况，则需要使用直方图和密度曲线图。下面分析某网站日 PV 数据的分布。

数据：	日期	页面浏览量（万）
	2010.1.1	254.99
	2010.1.2	245.39
	……	……

1. 使用直方图查看数据分布

页面浏览量 PV 是一个连续型变量，其范围约为 [220，280]。把这个范围以 3 为间隔进行分隔，统计 PV 在每段范围内的数目。代码如下：

```
par(mfrow=c(3,1))
hist(pv,main="页面浏览量的直方图：统计尺度间隔为 3",xlab="页面浏览量（万）",breaks=seq
(220,280,3))
```

然后再以 1 为间隔分割，同样统计其数目。代码如下：

```
hist(pv,main="页面浏览量的直方图：统计尺度间隔为 1",xlab="页面浏览量（万）",breaks=seq
(220,280,1))
```

上面的代码使用 hist 函数作为绘制直方图的核心函数，其中通过 breaks 参数可以设置被分析变量 pv 的统计尺度。例如，把 breaks 参数设置为 seq（220,280,3）向量，hist 函数会依次统计变量 pv 在 [220，223）[223，226）范围内的样本数目。

注意

直方图和柱状图在绘图中类似，但是它们表达的意义完全不同。柱状图中各个立柱的高度直接代表 pv 的取值，x 轴代表 pv 样本点序列号。而在直方图中，x 轴代表 pv 的统计尺度，各个立柱的高度表示各个统计尺度下包括 pv 样本的数目。

2. 使用密度曲线图查看数据分布

这里要使用到微积分的思想，当把统计尺度的间隔无限缩小时，直方图就变为了密度曲线图。密度曲线的绘制并无其他特殊函数，先通过 density 函数计算出变量的密度曲线数据，再使用 plot 函数直接绘制出其曲线图。

```
plot(density(pv),main="页面浏览量的密度曲线图",xlab="页面浏览量（万）")
rug(pv)
```

rug 是用于绘制坐标轴密度线的函数，具体可以参考 3.3.4 “坐标轴”。上面 3 段代码最终产生的图形如图 3-50 所示。

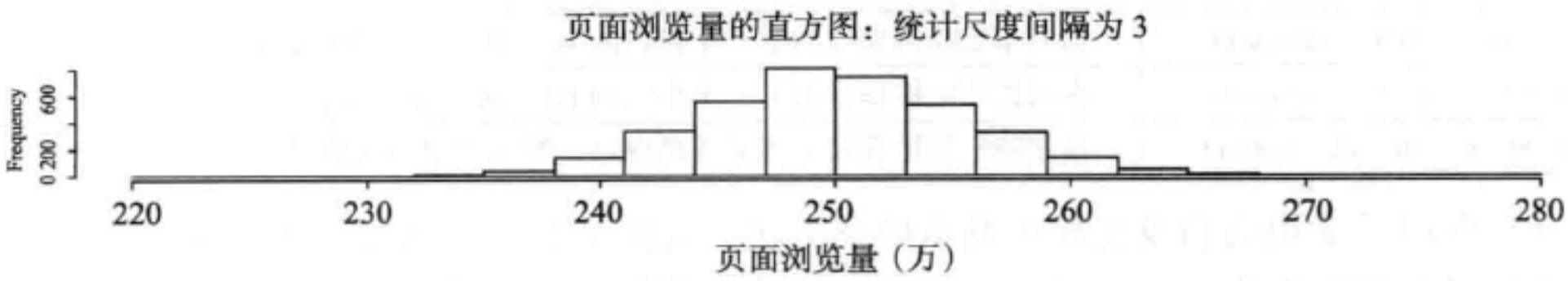


图 3-50 页面浏览量的密度曲线图

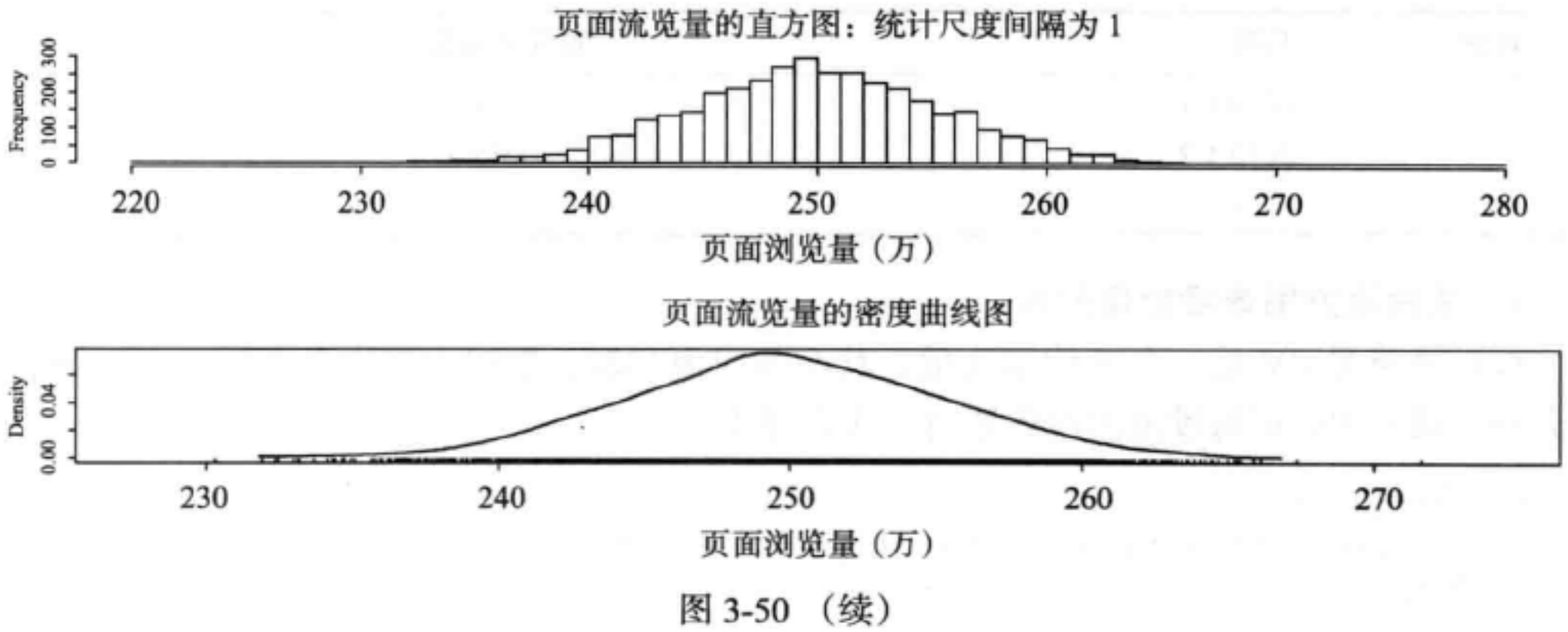


图 3-50 （续）

3.5 绘图窗口操作函数

3.5.1 单一窗口中的子绘图区域布局

par 函数中的 mfrow 参数可以把当前活动窗口均分割成子绘图区域。其取值是 c（nrow, ncol）形式，分别设置行数和列数。

在前面各个章节中已经多次使用 par（mfrow=c（<nrow>,<ncol>））形式设置子绘图区域，这里就不再详述了。

注意

par 函数不仅可以设置单一窗口内的多图布局，还可以统一设计其他常用的点、线、文字等元素的参数属性值。具体使用方式可以通过键入“?par”命令来查看。

3.5.2 绘图窗口操作

绘图窗口函数主要是指 dev.<xxx>() 的一系列函数，如表 3-17 所示。

表 3-17 绘图窗口函数

函数形式	说明
dev.new()	新建一个绘图窗口
dev.cur()	返回当前活动的绘图窗口 ID
dev.list()	返回所有已经打开的绘图窗口 ID
dev.off（which = dev.cur()）	关闭指定绘图窗口，默认关闭当前活动窗口
dev.next（which = dev.cur()）	返回指定绘图窗口的下一个窗口的 ID，默认值为当前活动窗口
dev.prev（which = dev.cur()）	返回指定绘图窗口的上一个窗口的 ID，默认值为当前活动窗口
dev.set（which = dev.next()）	指定特定绘图窗口为当前活动窗口，默认指定当前活动窗口的下一个窗口

以上介绍了 R 语言自身提供的基本绘图方法，实质上 R 语言还提供了很多的扩展绘图包，其中最为常用的是 ggplot2 和 lattice 包，本书就不详细介绍了。

第4章
单指标分析

本章是网站分析数学理论的第一个章节，这里先介绍单一指标变量，包括描述性分析、质量控制、样本差异对比，并在第9章中讲解样本间关联性的问题。

- 单指标描述性分析：是为了让分析者对指标数值有一个整体的感知，便于同其他行业网站或者网站内的其他分类情况进行同指标对比。例如，研究页面浏览量的平均数和数据分布、研究用户访问深度（用户每次访问网站时浏览的页面数）的众数和频分布等。
- 单指标质量控制：用于监控该指标的日常异常点，并了解短期波动趋势，便于及时发现网站的异常情况。例如，监控购物车转化率近2周内的波动和异常点。
- 单指标样本差异对比：用于衡量相同指标在不同情况下是否具有显著性的差异。本质上这已经不是单纯地研究单指标了，而是在考虑其他分类变量对该指标是否产生了显著的影响。例如，在对购物车功能进行改版优化后，对比购物车转化率数据，确定其是否有显著的差异，以衡量本次改版优化的效果。
- 单指标样本间关联性：针对点击流（序列关系的样本）模型的样本，找到样本间的关联性。例如，分析购物车内商品间的关联性，以找出可以交叉销售的商品。或者通过分析进入购物车页面所在的会话曾浏览过的页面，以找出那些促使用户发起添加购物车行为的页面。

4.1 指标描述

4.1.1 平均值和集中趋势

在3.4.9中曾提到了直方图的应用，如果对某指标样本的取值进行等分分割，产生多个子范围，然后统计大量样本取值在各个子范围内的数量，既可以分析该指标样本数据的频数，又可以绘制出直方图（见图4-1），以便进一步分析。

一旦呈现如图 4-1 所示的形态，就可以看出，大部分样本的取值较为集中，只有少部分分散在相对范围较远的区域中。这种形态符合正态分布（正态分布的定义已在 2.10.5 节中介绍，在 4.1.2 节中会详细讲解如何判定一个指标服从正态分布）。此时可以使用样本的平均数（或称为均值）来描述该指标，其定义为：

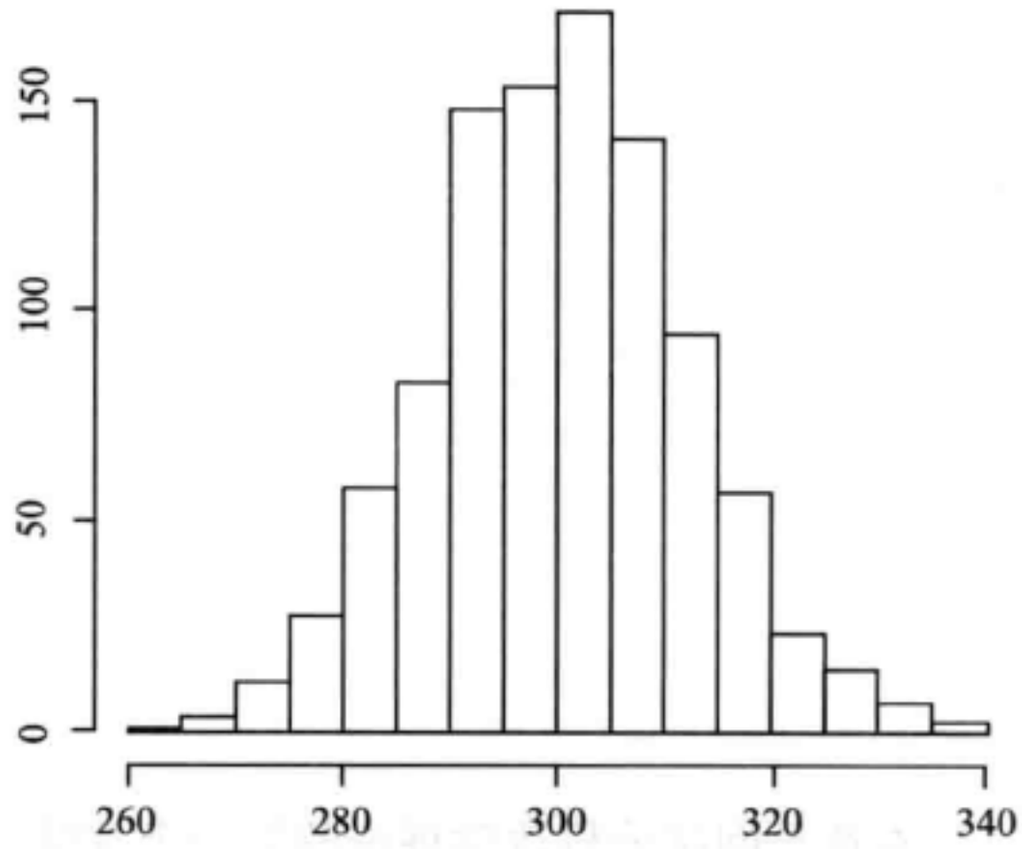


图 4-1　直方图示例

$$\overline{x}=\frac{1}{n}\sum_{i=1}^{n}x_i$$

其中，$\overline{x}$是样本平均数，n 是样本数，x_i 是每个样本的数值。

在 R 语言中使用 mean（x, trim=0, na.rm=FALSE）来计算均值。其中 x 是向量（或矩阵 / 数组 / 数据框），它涵盖了样本的数据信息。trim 是对异常点样本做预处理的方法，其取值范围是 [0,0.5]，其含义是：先把左右两侧 trim 比例数的边缘（最大和最小的）样本剔除，然后计算样本平均值。在默认情况下，其取值为 0，表示不做任何处理。

```
data<-c(1,30,9,8,99,23,91,30,40,234,17)
mean(data,trim=0.1)
[1] 38.55556
mean(sort(data)[2:10])
[1] 38.55556
```

在上述代码中，trim 设置为 0.1，样本数据 data 的长度为 11，可见，需要剔除 11×0.1=1 个最小样本和 1 个最大样本，剔除后的样本序列也可以使用 sort（data）[2:10] 来表示。因此 mean（data, trim=0.1）和 mean（sort（data）[2:10]）最终的结果一致。

na.rm 参数用于设置对默认值 NA 的处理方式：取值为 TRUE 时，允许样本中包含 NA，否则不允许包含。代码如下：

```
data<-c(1,30,9,8,99,23,91,NA,40,234,17)
mean(data)
[1] NA
mean(data,na.rm=T)
[1] 55.2
```

默认情况下，na.rm=FALSE，即在计算平均值时不允许有 NA，一旦包括，就返回 NA。na.rm 在被设置为 TRUE 后，计算样本均值前会先剔除 NA 样本，这样就返回了正常的平均值。

1. 实例：用户的月消费金额

上面提到了，如果指标符合集中趋势（正态分布），则可以使用平均值来描述该指标。

但是如果该指标的直方图是如图 4-2 所示的形式呢?

图 4-2 选取了某电商网站中 750 名女性会员和 250 名男性会员的每月消费金额作为数据。代码如下:

```
hist(data,main="用户每月消费金额的直方图",xlab="每月消费金额(元)",ylab="样本数")
```

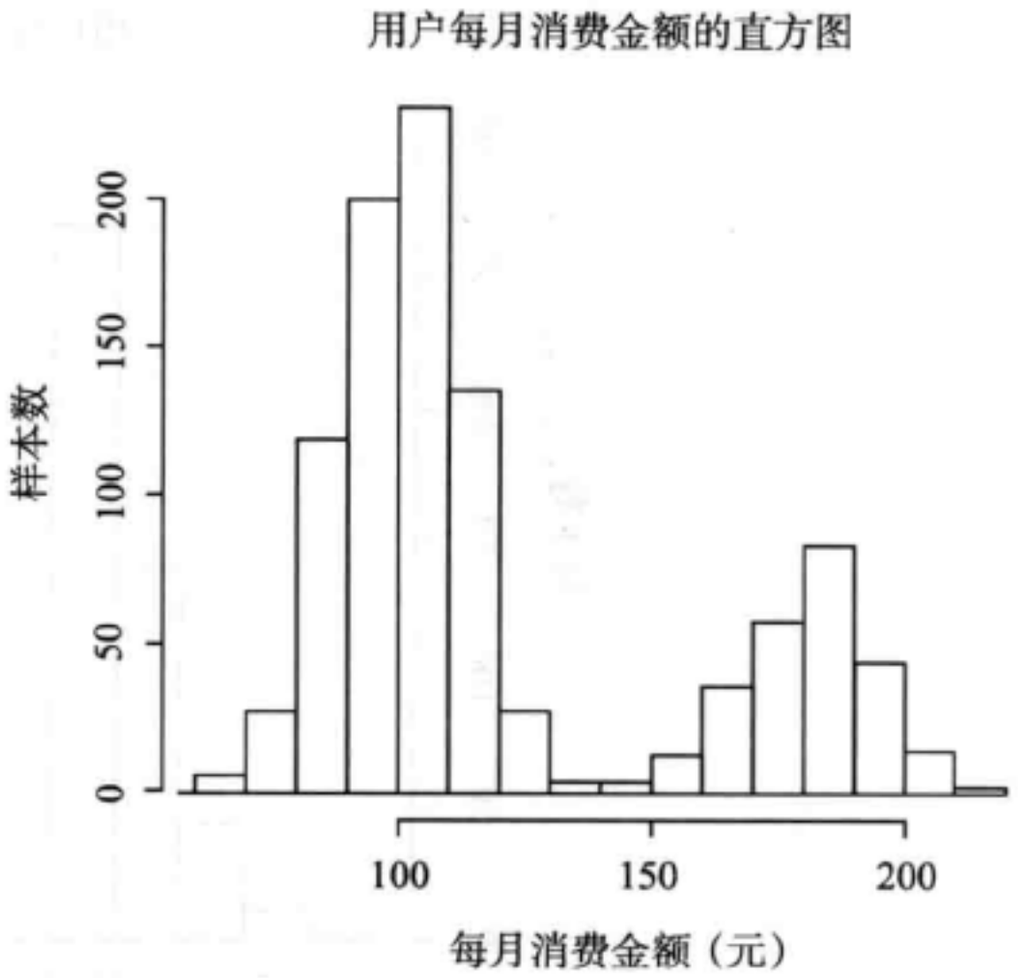


图 4-2 月消费金额直方图

从图 4-2 可以看出,用户的月消费金额明显有两个集中趋势,分别在 100 和 170 左右。而通过计算得出的平均值却是 120。如果使用 120 代表该指标就会出现误差,因为每月消费在 115 ~ 125 元的用户数只有 40 人左右,只能代表 4% 的用户。那么,这种情况下该如何处理呢?此时可以猜测性别这个分类变量是否会影响该指标。如果依据性别变量区分原始样本,然后分别绘制其直方图,问题就有了答案。代码如下:

```
par(mfrow=c(1,2))
hist(data.man,main="男性用户每月消费金额的直方图",xlab="每月消费金额(元)",ylab="样本数")
hist(data.woman,main="女性用户每月消费金额的直方图",xlab="每月消费金额(元)",ylab="样本数")
```

绘制结果如图 4-3 所示。

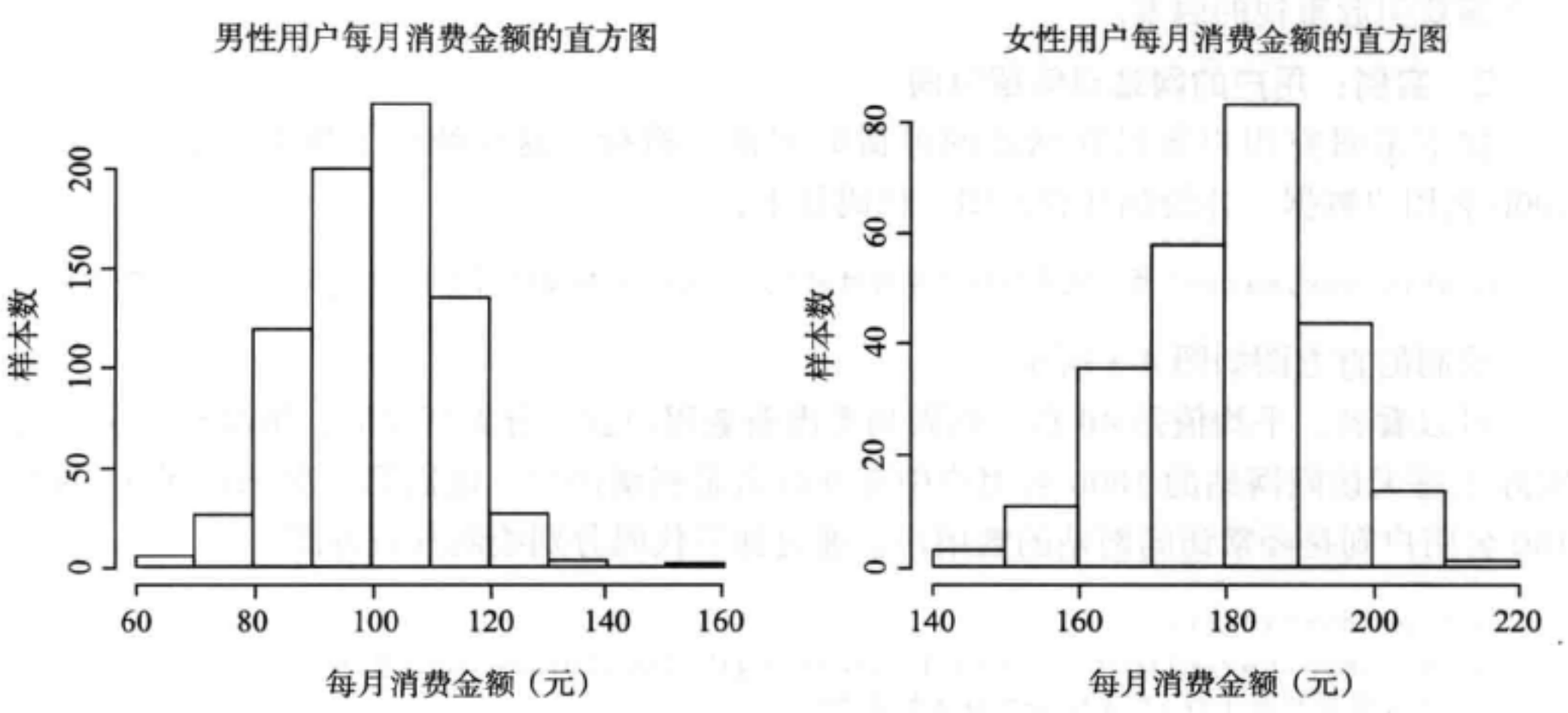


图 4-3 男性女性月消费金额直方图

由图 4-3 可以清楚地看见男性每月消费金额趋于 100 元 / 月,而女性则趋于 180 元 / 月。

图 4-2 所使用的样本还不是十分具有迷惑性，如果在研究网站月消费金额时将样本数量改为：男性样本选取为 975 个，女性样本只有 25 个，那么此时的直方图如图 4-4 所示。

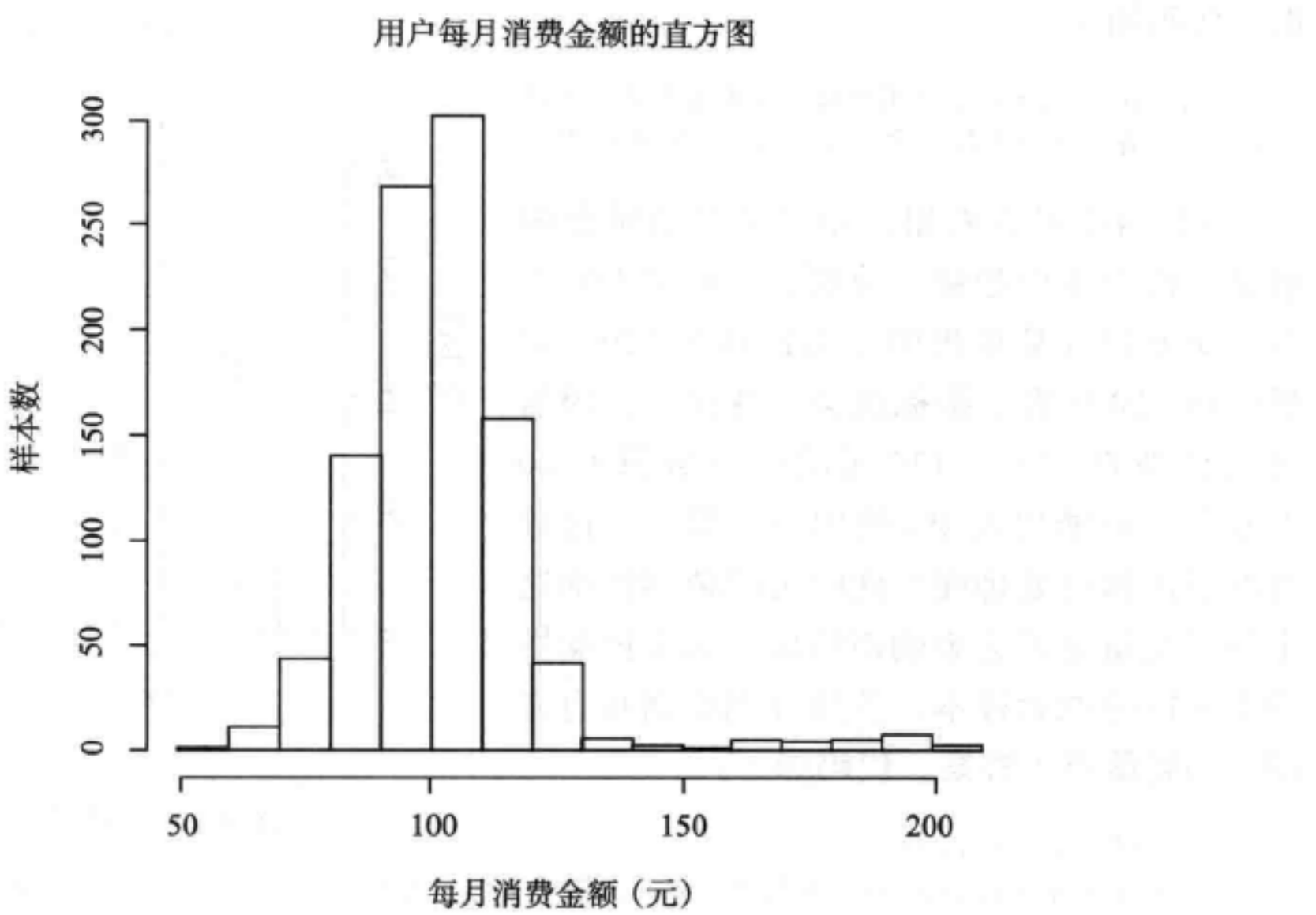


图 4-4 结果展示

由于女性样本极少，以至于 170 元 / 月的女性消费集中趋势没有很明显地展示出来。这时分析人员就会错误地以其平均值 100 作为所有用户的月均消费金额。可见样本的质量也是一个需要引起重视的因素。

2. 实例：用户的网站日停留时间

接下来研究用户每日在网站的停留时间这一指标。这里统计了所有当日访问网站的 1000 名用户数据，并绘制其直方图。代码如下：

```
hist(times,main="用户在网站的日停留时间",xlab="日停留时间（秒）",ylab="样本数目")
```

绘制的直方图如图 4-5 所示。

可以看到，平均值是 40 秒，然而当考虑新老用户这一分类变量后，情况却完全不同了。实际上每天访问网站的 1000 名用户中有 900 名是新增用户，他们第一次来到网站。剩余的 100 名用户则是经常访问网站的老用户。通过如下代码分别绘制其直方图。

```
par(mfrow=c(1,2))
hist(times.new,xlim=c(0,160),breaks=seq(0,160,10),main="新用户在网站的日停留时间",xlab="日停留时间（秒）",ylab="样本数目")
hist(times.old,main="老用户在网站的日停留时间",xlab="日停留时间（秒）",ylab="样本数目")
```

绘制结果如图 4-6 所示。

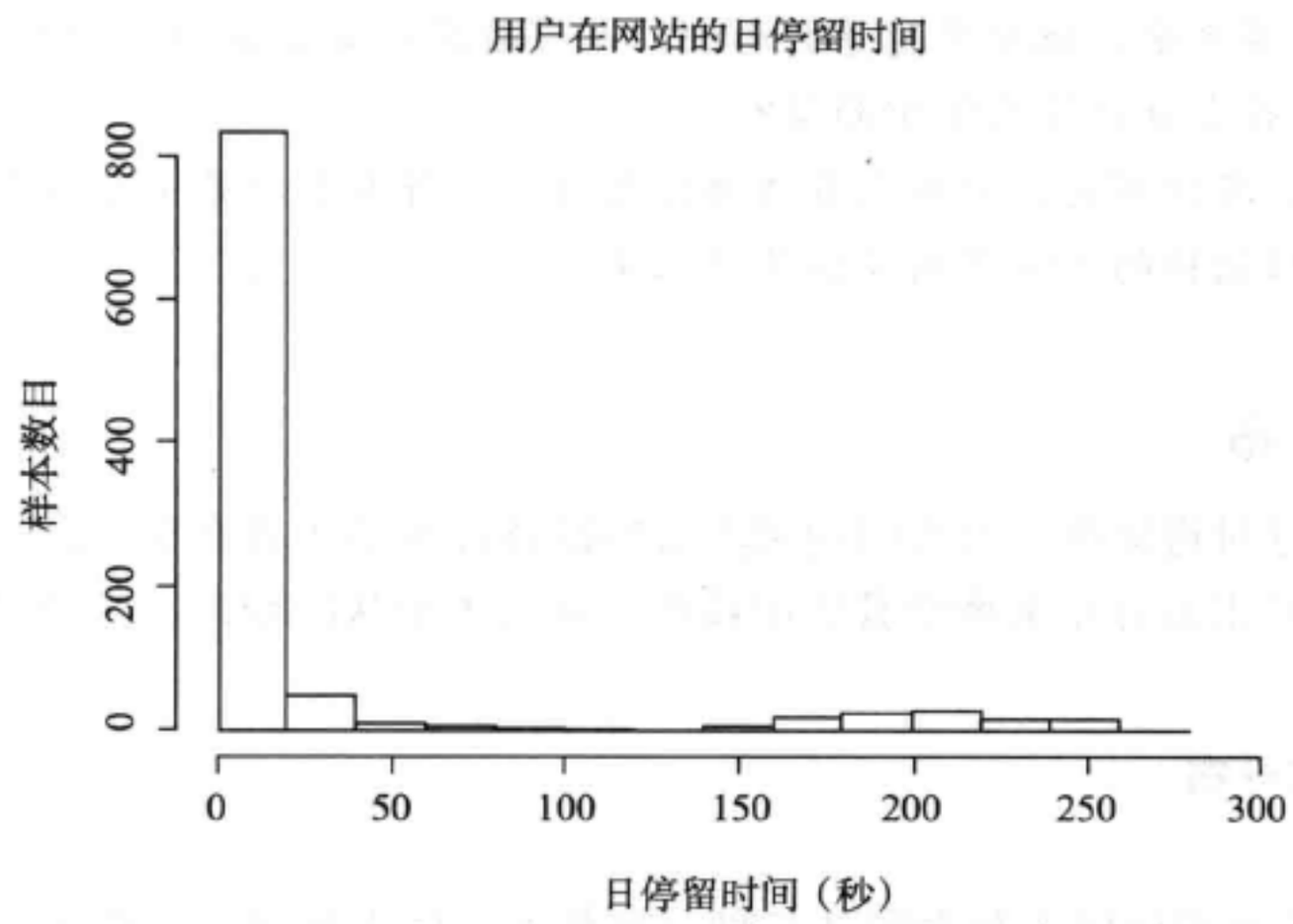


图 4-5 用户日停留时间

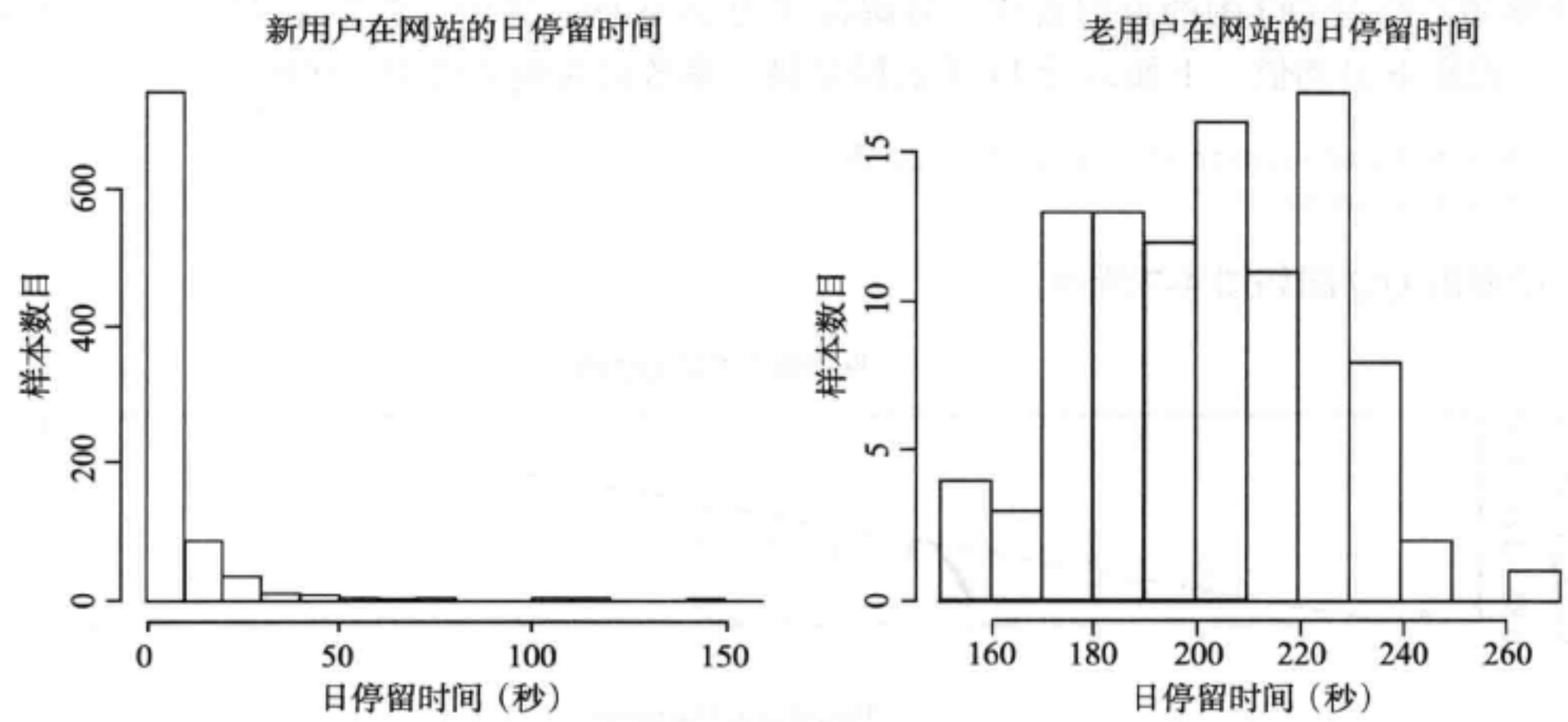


图 4-6 新老用户的日停留时间

可见对于新用户，其停留时间呈指数下降，并集中在 1 ～ 15 秒钟，平均值就是 10 秒。而对于老用户，则分散在 170 ～ 240 秒，虽然没有很明显的集中趋势，但是仍和新用户的指标数据有明显的不同。

注意

在分析网站数据时，平均数虽然是最常见的衡量指标，但是我们往往会忽略一个前提，那就是该指标的样本取值是有明显的集中趋势或者说该指标是服从正态分布的。因此在使用一个平均数来衡量一个指标前，要先绘制其样本的直方图，并检验其是否符合正态分布。

一旦不符合正态分布，则需继续思考以下两个问题：

1）该指标是否会受其他分类变量的影响，而导致其不具有集中趋势？

2）该指标是否本身就没有集中趋势？

即使是符合正态分布的，分析人员仍要提醒自己：样本本身是否会存在问题，从而导致一些分类变量对该指标的影响作用没能体现出来？

4.1.2 正态分布

以上只能通过对指标样本直方图的观察来判断该样本是否具有集中趋势，下面通过检验指标样本是否服从正态分布来衡量其集中趋势。并对不服从的变量进行处理，使其服从正态分布。

1. 检验正态分布

（1）QQ 图

QQ 图用于检验指标样本是否服从某种分布状态，样本越是服从某种分布规则，绘制的 QQ 图越会趋于一条直线。在 R 语言中使用 qqnorm 函数来绘制正态分布的 QQ 图。qqline 函数则用于绘制 QQ 图的近似直线，其解析式为 y=ax+b。其中，斜率 a 是正态分布中的标准差，截距 b 是均值。下面以近 14 天的网站跳出率数据为例来说明。代码如下：

```
qqnorm(bounce,main=" 网站跳出率的 QQ 图 ")
qqline(bounce)
```

绘制的 QQ 图如图 4-7 所示。

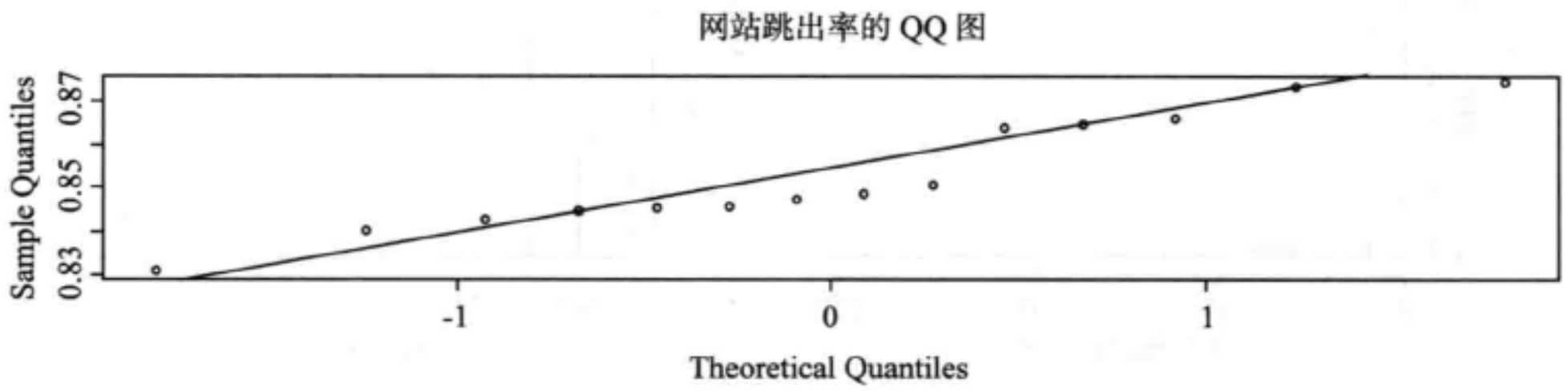


图 4-7 QQ 图

由于 QQ 图中的各个散点趋近于一条直线，因此可以初步判定该网站的跳出率服从正态分布。

（2）shapiro.test 函数

不论是直方图，还是 QQ 图，都是通过图形的方式检验样本是否服从正态分布。而 shapiro.test 函数则可以通过量化来判定这个问题。代码如下：

```
sol<-shapiro.test(bounce)
sol
        Shapiro-Wilk normality test
data:  bounce
```

```
W = 0.9194, p-value = 0.2158
sol$p.value
[1] 0.2157997
```

在上面的代码中，sol 为 shapiro.test 函数返回的对象，如果 sol$p.value 大于显著性水平 0.05，则可以说该样本向量服从正态分布，并且该数值越接近于 1 越好。

（3）自定义检验服从正态分布的函数

下面自定义一个函数来判定样本是否服从正态分布，它结合了上面提到的 QQ 图和 shapiro.test 函数两种方式。代码如下：

```
norm.test<-function(input.data,alpha=0.05,pic=TRUE){
    if(pic==TRUE){# 画图形
        dev.new()
        par(mfrow=c(2,1))
        qqnorm(input.data,main="qq 图 ")
        qqline(input.data)
        hist(input.data,freq=F,main=" 直方图和密度估计曲线 ")
        lines(density(input.data),col="blue")# 密度估计曲线
        x<-c(round(min(input.data)):round(max(input.data)))
        lines(x,dnorm(x,mean(input.data),sd(input.data)),col="red")# 正态分布曲线
    }
    sol<-shapiro.test(input.data)
    if(sol$p.value>alpha){
        print(paste("success: 服从正态分布，p.value=",sol$p.value,">",alpha))
    }else{
        print(paste("error: 不服从正态分布，p.value=",sol$p.value,"<=",alpha))
    }
    sol
}
```

在自定义函数 norm.test 中，input.data 参数是样本向量。alpha 参数是显著性水平，默认选择为 0.05，在内部代码中，用它来作为 shapiro.test 函数进行正态分布检验时的标准，该值设置得越高（最大值为 1），检验的门槛也就越高。pic 参数用于设置是否绘制出 QQ 图和直方图，以及概率密度曲线。判定其服从正态分布后，结果提示语为：

“success: 服从正态分布，p.value=…>alpha”。

如果判定为不服从，则结果提示语为：

“error: 不服从正态分布，p.value=…<=alpha”。

下面使用上面的自定义函数检验某网站近 2 个月的跳出率指标。代码如下：

```
> norm.test(bounce)
 [1] "success: 服从正态分布，p.value= 0.0708652334811504 > 0.05"
        Shapiro-Wilk normality test
data:  input.data
W = 0.9636, p-value = 0.07087
```

这里使用常规的检验门槛（显著水平）0.05，并输出如图 4-8 所示的 QQ 图和直方图。

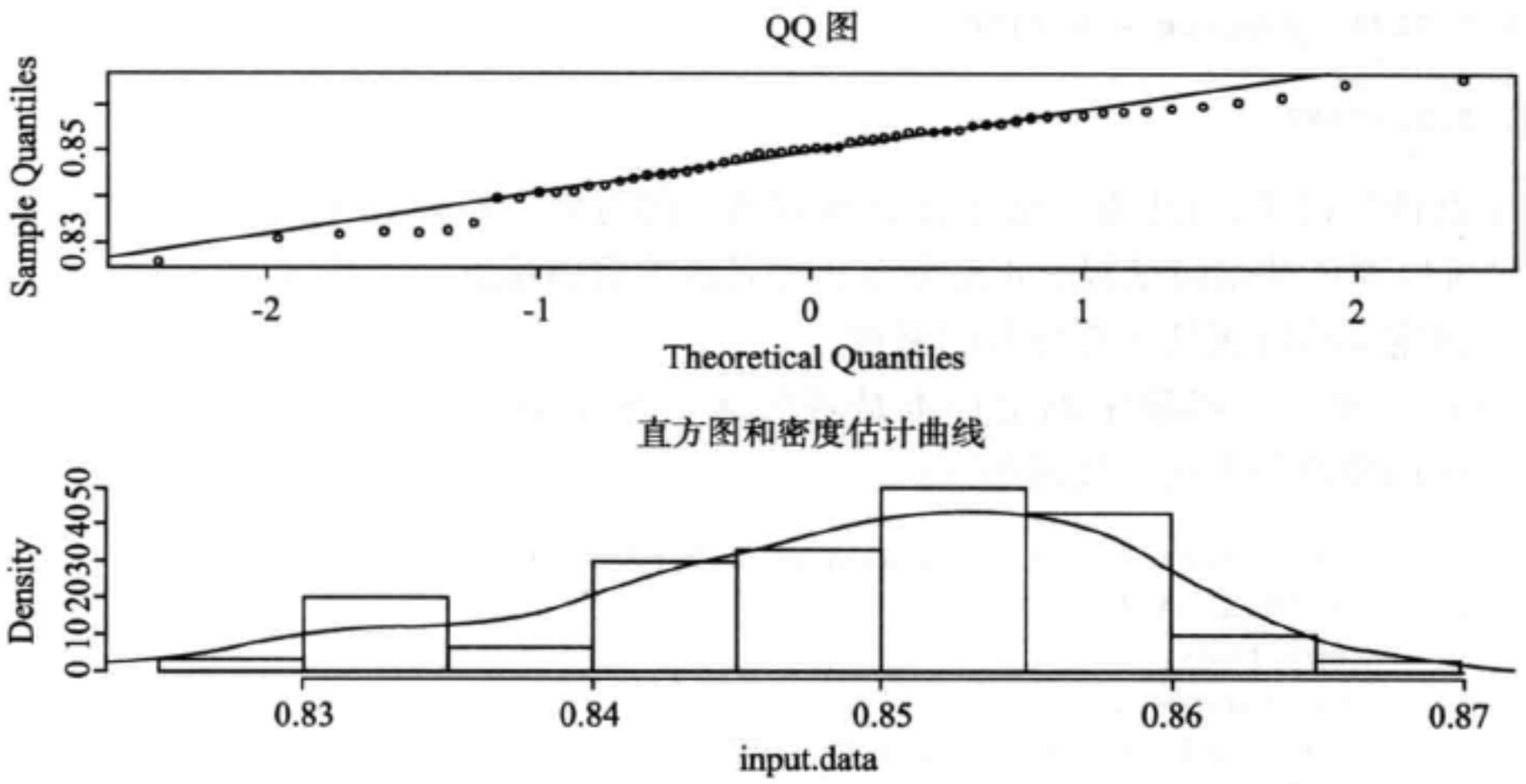


图 4-8 QQ 图和直方图

2. 正态化处理

在很多数据分析的算法中，都要求被处理的数据服从正态分布。因此如果一个指标不服从正态分布，那么对其进行分析的数学方法就变得很有限了。接下来探讨如何处理非正态化指标的正态化。

（1）差分处理 diff

先绘制上面实例（某网站近 2 个月的跳出率指标）中的跳出率散点图，代码如下：

```
plot(bounce,main="近 2 个月网站跳出率的散点图",ylab="网站跳出率",xlab="日期",pch=16)
abline(h=mean(bounce))
```

绘制结果如图 4-9 所示。

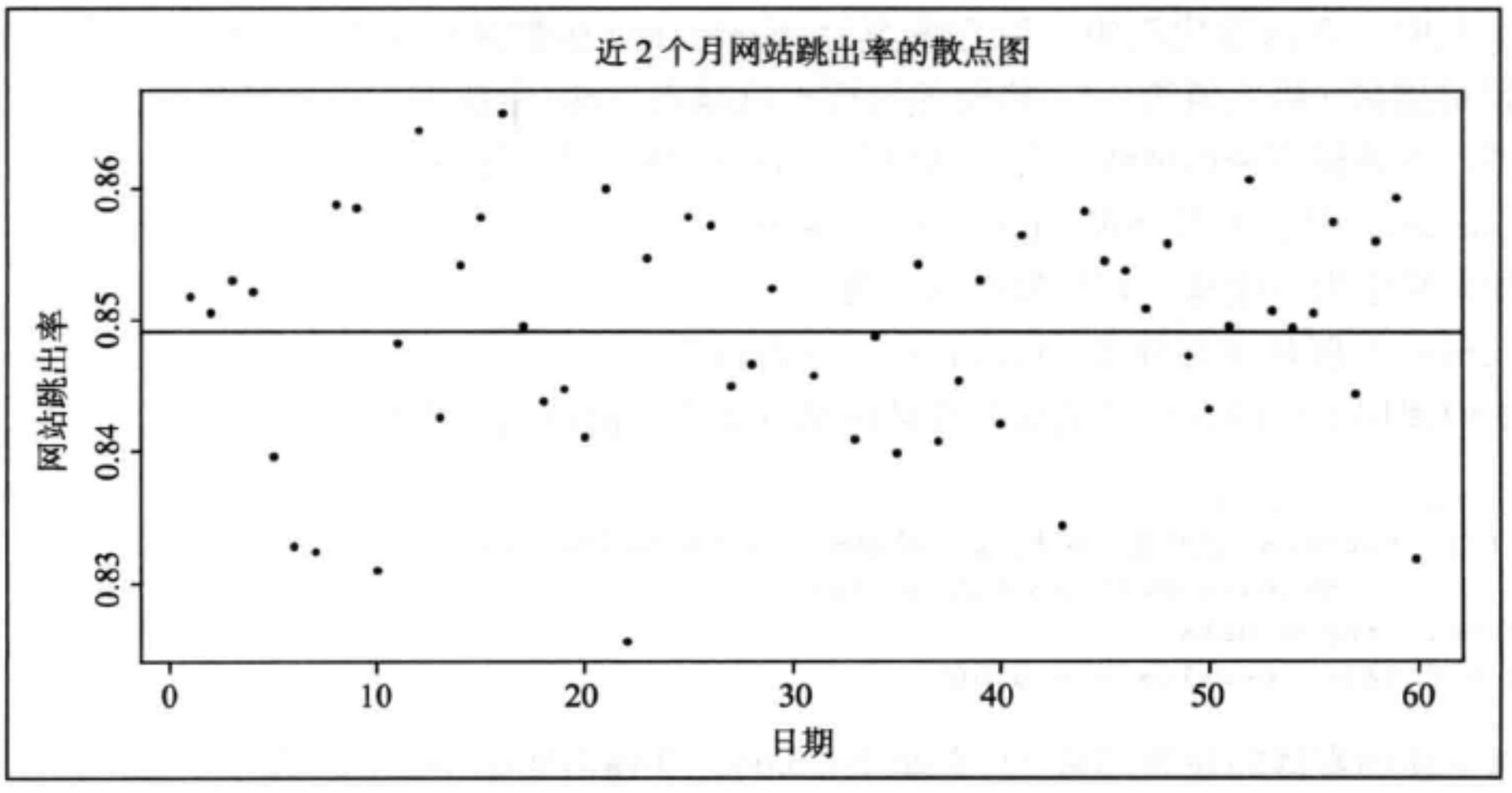


图 4-9 近 2 个月的跳出率散点图

可见跳出率是围绕样本均值上下随机分布的，并且不会随着时间的改变而显示出明显的上升或下降趋势。然而如果随着网站的不断优化，其跳出率在近两个月中呈逐步下降趋势，那么代表什么呢？先来看看其散点图，如图 4-10 所示。

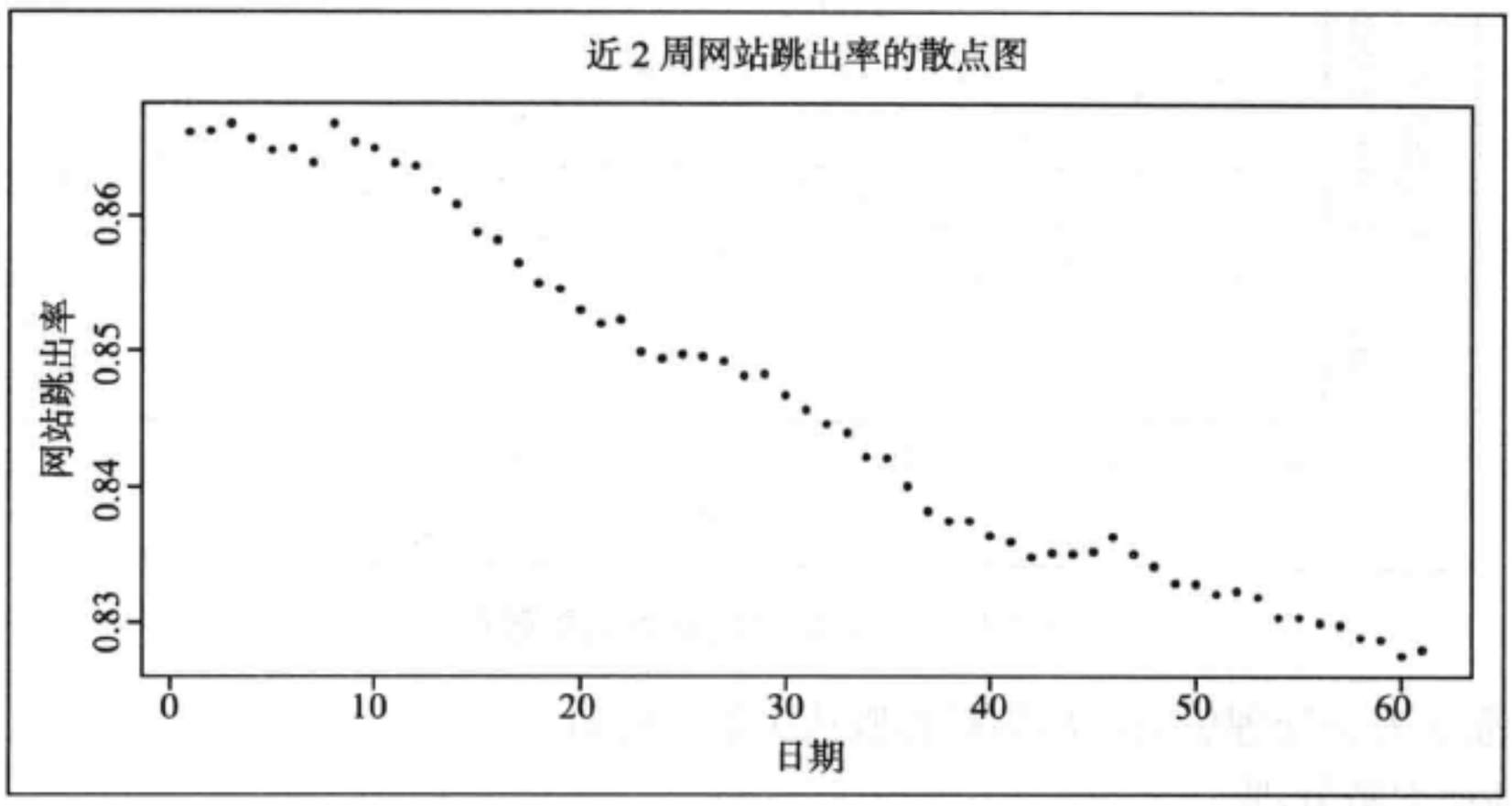


图 4-10 网站不断优化后的跳出率散点图

由图 4-10 可观察到，在时间序列的散点图中，跳出率是趋于一条下滑斜线的，并随时间变化有明显的下降趋势。再使用 norm.test 函数判断，其结果为

```
norm.test(bounce2)
[1] "error: 不服从正态分布, p.value= 0.0150443284434951 <= 0.05"
        Shapiro-Wilk normality test
data:  input.data
W = 0.9504, p-value = 0.01504
```

经过 norm.test 函数可判断出，它并不服从正态分布。此时应如何分析和监控这两个月内的跳出率指标呢？

假设上面提到如图 4-10 所示的散点图趋于直线的解析式是 bounce[i]=a*t[i]+b。则 bounce[i+1]-bounce[i]=a，即跳出率的差分（后一项减前一项）并不随时间序列而变，可以预料到，在对跳出率做差分处理后，其很有可能会服从正态分布（主要看散点图和直线的拟合程度）。经过差分计算后，跳出率指标的散点图如图 4-11 所示。

它和图 4-10 很类似，也是随机分布于均值上下的，并且不随时间有明显的上升或下降趋势。使用 norm.test 函数来检验：

```
norm.test(diff(bounce2))
[1] "success: 服从正态分布, p.value= 0.757168242563001 > 0.05"
        Shapiro-Wilk normality test
data:  input.data
W = 0.9867, p-value = 0.7572
```

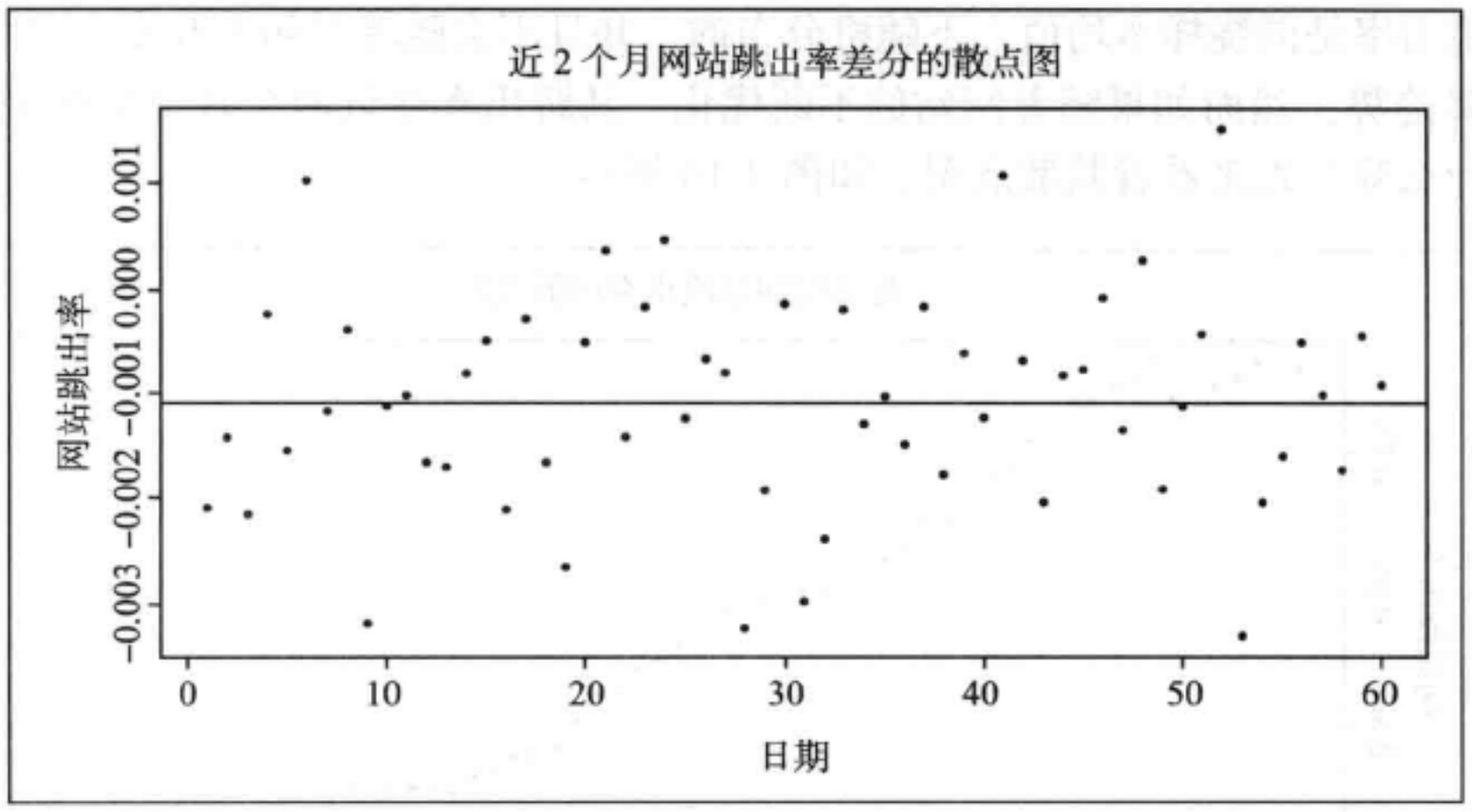


图 4-11 差分处理后的跳出率散点图

结果显示差分处理后的跳出率指标服从了正态分布。

（2）log 对数处理

如果网站的跳出率数据不是呈线性逐步下降，而是呈对数急剧下降，那么可以使用 log 函数对其进行对数处理。

4.1.3 频数分析

当某一指标不符合正态分布，甚至连集中趋势都不明显时，该如何分析相关数据呢？此时只能回归到使用频数分析来确定不同范围下样本的比例。下面的代码用于研究登录用户的网站停留时间指标。这里把 times 分为：" 小于 10 秒 "、"10-30 秒 "、"30-60 秒 "、"1-3 分 "、"3-10 分 "、"10-30 分 "、" 大于 30 分 " 等 7 个子范围，再逐一统计各子范围内的样本数量。由于并未使用平均分割的方法来划分子范围，所以使用 hist 直方图绘制出的图形很不规范，这里使用另一种方式：barplot 柱状图函数。代码如下：

```
    times<-cut(times,breaks=c(0,10,30,60,180,600,1800,max(times)))
    times<-table(times)
    barplot(times,main=" 登录用户日网站停留时间 ",xaxt="n")
    axis(side=1,at=seq(from=0.7,by=1.2,length.out=7),labels=c(" 小　于 10 秒 ","10-30 秒
","30-60 秒 ","1-3 分 ","3-10 分 ","10-30 分 "," 大于 30 分 "))
```

首先使用 cut 函数依据子范围（" 小于 10 秒 "、"10-30 秒 "、"30-60 秒 "……）把 times 指标转化为分类变量。然后使用 table 函数统计各子范围内样本的数目。最后使用 barplot 函数绘制柱状图，如图 4-12 所示。

虽然登录用户的停留时间不再具有集中趋势，但仍可以通过频数分析来获取重要的信息。例如，如果把停留 1 ～ 60 秒的会员称为非活跃会员，把停留 1 ～ 10 分钟的会员称为活跃用户会员，10 分钟以上的称为发烧友，那么，通过图 4-12 即可获知不同级别用户的比例和数量。

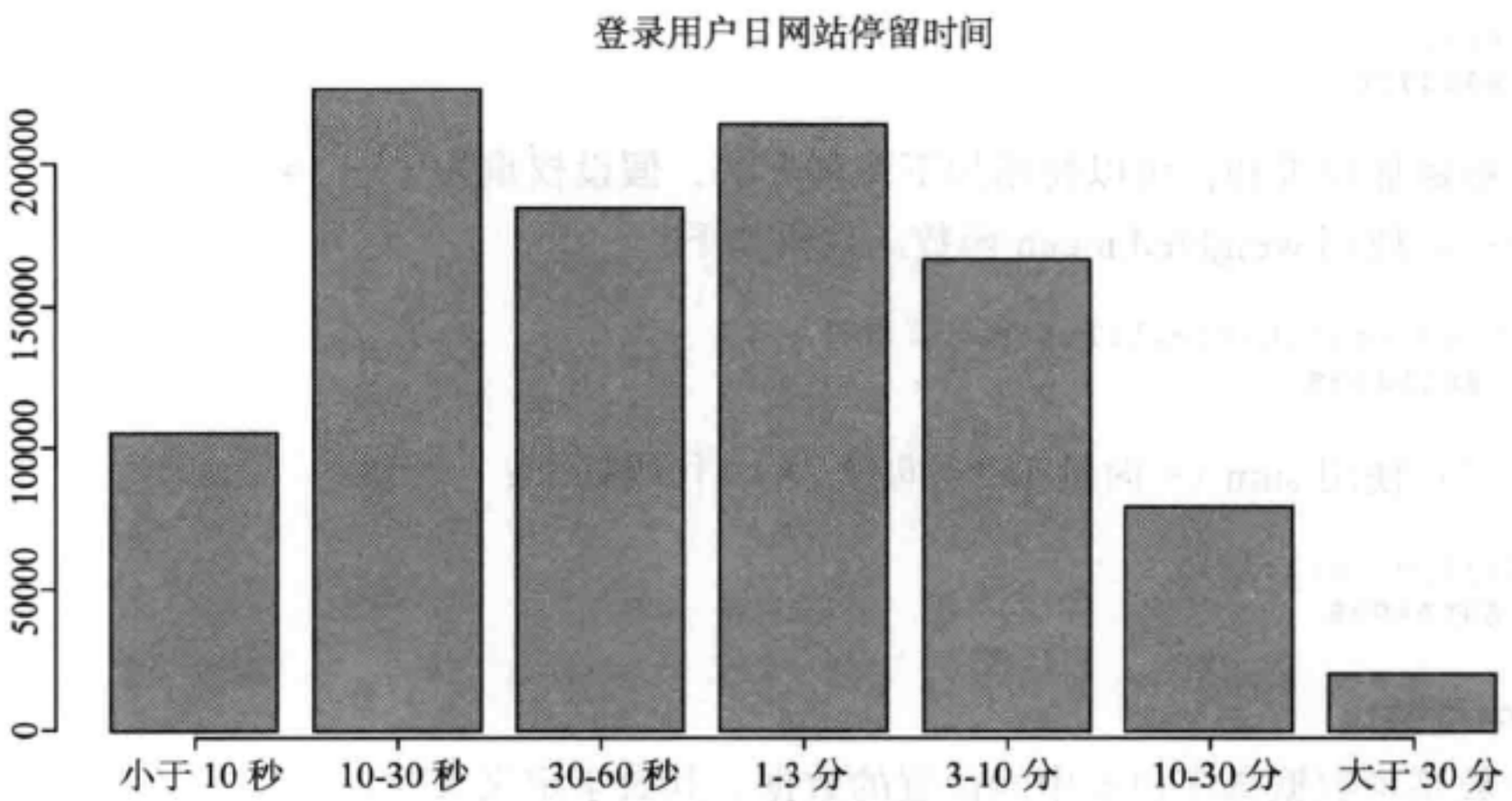


图 4-12 登录用户日停留时间

以上是对连续型指标的频数分析，在 4.4.1 节中还会详细介绍如何对分类型指标进行频数分析。

4.1.4 描述性分析指标

下面介绍连续变量的中心趋势和离散趋势。以某网站 14 天的基础数据为例，如图 4-13 所示。

该数据存储于数据框类型对象 web 中，其中 pv 列表示网站页面浏览量，uv 列表示独立访问用户数，v 列表示访问次数，bounce 列表示网站跳出率。

date	pv	uv	v	bce
2013.01.14	532251	223065	389721	0.8661
2013.01.15	574540	216166	398998	0.8745
2013.01.16	537528	217605	396596	0.8734
2013.01.17	593018	226722	425834	0.8639
2013.01.18	759865	258608	523487	0.8650
2013.01.19	843321	319029	580284	0.8459
2013.01.20	785337	296505	530150	0.8426
2013.01.21	637756	239219	426349	0.8448
2013.01.22	627096	237991	425332	0.8456
2013.01.23	618487	247237	435509	0.8487
2013.01.24	644126	260646	447280	0.8505
2013.01.25	535169	233432	354867	0.8308
2013.01.26	677159	280163	454841	0.8401
2013.01.27	635077	269671	436527	0.8476

图 4-13 数据集展示

1. 平均数

前面详细介绍了均值函数 mean，与之相关的 weighted.mean 函数可以计算加权处理后的平均值。公式如下：

$$\bar{x}=\sum_{i=1}^{n} wi*xi/\sum_{i=1}^{n} wi$$

代码如下：

```
weighted.mean(kj$pv,1:14)
[1] 649094.6
```

对于 pv 数值来说，离它越近的数据越有价值，这里由近到远线性地设置 14 条 pv 数据的权重。

2. 样本求和

在 R 语言中使用 sum 函数计算样本求和。例如，计算 14 天的 pv 指标总和的代码如下：

```
sum(kj$pv)
[1] 9000730
```

对指标做加权求和，可以使用如下两种方法，假设权重为 1 ～ 14。

方法一：使用 weighted.mean 函数。代码如下：

```
weighted.mean(kj$pv,1:14)*sum(1:14)
 [1] 68154938
```

方法二：使用 sum（< 向量 1>*< 向量 2>），代码如下：

```
sum(kj$pv *c(1:14))
[1] 68154938
```

3. 中位数

中位数是指数据排序位于中间位置的数据，其数学定义式为：

$$m_e=\begin{cases} x\left(\frac{n+1}{2}\right), & \text{当 } n \text{ 为奇数时} \\ \frac{1}{2}\left(x\left(\frac{n}{2}\right)+x\left(\frac{n}{2}+1\right)\right), & \text{当 } n \text{ 为奇数时} \end{cases}$$

如果数据的分布呈对称形态，则中位数与平均数很相近。中位数的一个特点是，不易受异常大或异常小的影响。在 R 语言中会使用 median 函数来计算中位数。代码如下：

```
median(kj$pv)
[1] 631086.5
```

由于 pv 向量的长度是 14，所以 median（kj$pv）实际也可以使用 sum（sort（kj$pv）[7:8]）/2 来计算。其中，sort 是将由向量由小到大排序的函数。

4. 极差和五数总括

min 是计算最小值的函数，max 是计算最大值的函数。其极差为最大值减最小值。代码如下：

```
max(kj$pv)
[1] 843321
min(kj$pv)
[1] 532251
max(kj$pv)-min(kj$pv)
[1] 311070
```

计算指标的中位数、上 / 下四分位，以及最大 / 最小值这 5 个数据叫五数总括。在 R 语言中使用 fivenum 函数来计算这 5 个数据。代码如下：

```
fivenum(kj$pv)
[1] 532251.0    574540.0    631086.5    677159.0    843321.0
```

5. 百分位数

百分位数是中位数和五数总括的扩展，将指标样本由小到大进行排序，当 0<=p<1 时，

p 分位点定义为：

$$m_p=\begin{cases} x\big([np]+1\big), & \text{当 } np \text{ 不是整数时} \\ \frac{1}{2}\big(x(np)+x(np+1)\big), & \text{当 } np \text{ 是整数时} \end{cases}$$

其中 [np] 表示 np 的整数部分。在 R 语言中使用 quantile 函数计算百分位数，默认情况下计算 0%、25%、50%、75%、100% 这 5 个百分位数。代码如下：

```
quantile(kj$pv)
   0%        25%        50%        75%       100%
532251.0   579159.5   631086.5   668900.8   843321.0
```

如果想计算 0%、20%、40%、60%、80%、100% 的百分位数据，则可以通过设置 probs 参数来完成。其代码如下：

```
quantile(kj$pv,probs=seq(0,1,0.2))
   0%        20%        40%        60%        80%       100%
532251.0   559735.2   620208.8   637220.2   710241.4   843321.0
```

6. 方差和标准差

方差和标准差可衡量指标样本的离散趋势。

样本方差的数学定义为：

$$s^2=\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2$$

其含义是样本相对于样本均值的偏差平方和的平均值（注意在计算平均时，不是使用 n，而是使用 $n-1$）。下面使用 var 函数计算 pv 指标的方差。代码如下：

```
var(kj$pv)
[1] 9104616993
```

标准差是方差的正平方根，其公式如下：

$$s=\sqrt{s^2}=\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2}$$

下面使用 sd 函数来计算 pv 指标的标准差。代码如下：

```
sd(kj$pv)
[1] 95418.12
```

4.2 异常点监控

4.2.1 概述

异常点监控的前提：指标并无明显的趋势，既不是持续上涨，也不是下降，同时不会经常出现大幅波动，即服从正态分布。

1. 异常点的定义

假设指标服从均值 μ 和标准差 σ 的正态分布，如图 4-14 所示。

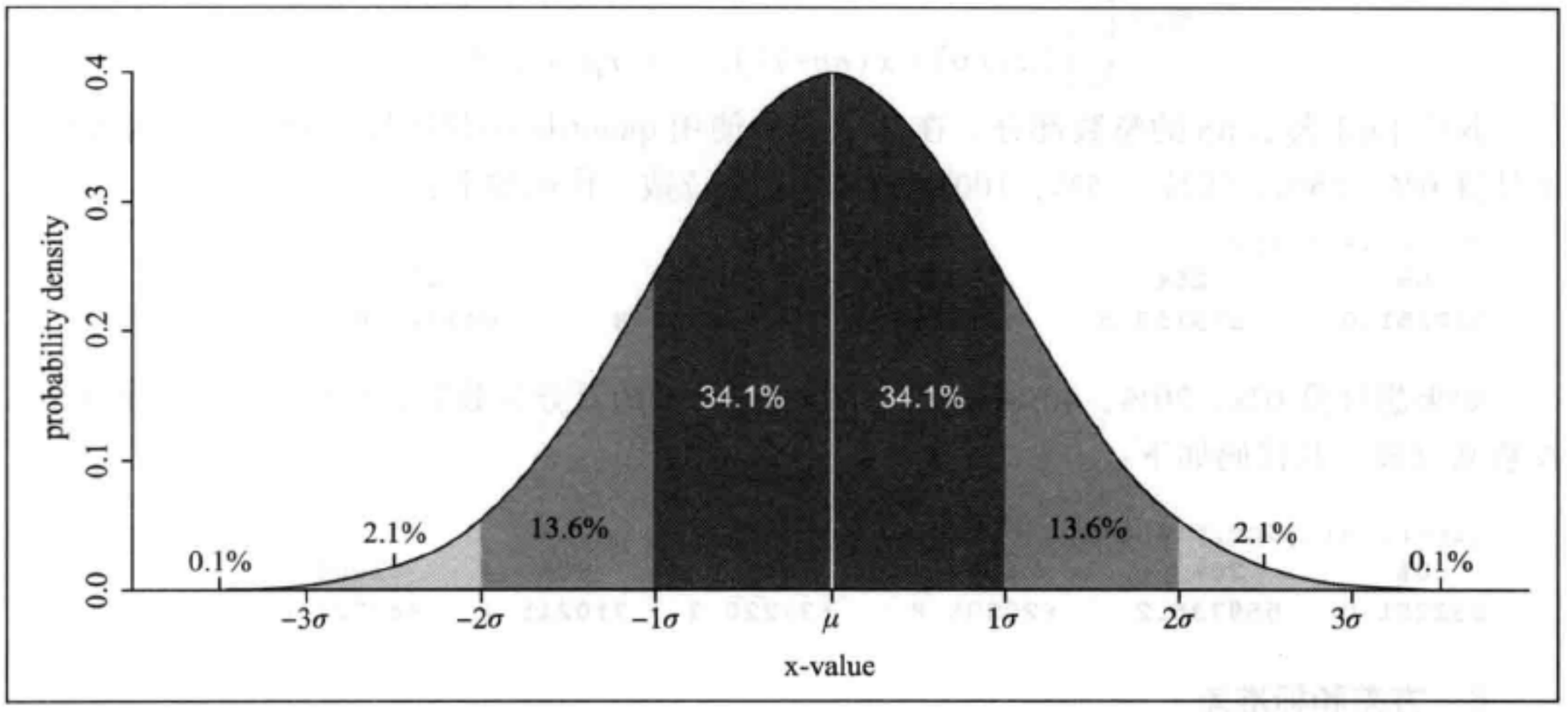


图 4-14　正态分布图

当指标 x 属于 $(\mu-\sigma, \mu+\sigma)$ 时，对应的概率（和 x 轴间的面积）是 $2 \times 34.13\%=68.26\%$。

当指标 x 属于 $(\mu-2\sigma, \mu+2\sigma)$ 时，对应的概率是 $68.26\%+2 \times 13.6\%=95.46\%$。

当指标 x 属于 $(\mu-3\sigma, \mu+3\sigma)$ 时，对应的概率是 $95.46\%+2 \times 2.14\%=99.74\%$。而处于（负无穷大，$\mu-3\sigma$] 和 [$\mu+3\sigma$, 正无穷）范围时，样本的概率为 0.26%，这是一个小概率事件，我们称其为 3 倍标准差下的异常点，并分别把 $\mu-3\sigma$ 和 $\mu+3\sigma$ 称为 3 倍标准差下的下限（LCL）和上限（UCL）。

除了使用 3 倍 σ 计算上下限数值外，还可以自定义标准来区分异常点。质量控制图的实质就是依据监控指标的样本数据来计算出 CL、UCL、LCL。

2. 质量监控图的使用

在图 4-15 中，通过 UCL 上限和 LCL 下限两条水平线把整个平面划分为绿色和橘红色区域。

如果把指标数据 – 序列号组成的散点图绘制于图 4-15 中，则[⊖]：

- ❑ 落于绿色区域的样本点为正常样本，落于橘红色区域的样本就是异常点样本。
- ❑ 如果近期的 3 个点中有 2 个点都高于 2 倍标准差下的 UCL 或都低于 2 倍标准差下的 LCL，或近期的 5 个点中有 4 个点都高于单倍标准差下的 UCL 或都低于单倍标准差下的 LCL，则称该指标有出现异常的趋势。
- ❑ 如果连续的 8 个点均高于或低于中心线，则称其有偏向性。
- ❑ 如果连续的 6 个点呈上升或下降趋势，则称其有明显的偏向趋势。
- ❑ 如果连续的 14 个点在中心线的上下呈交替状态，则称其具有周期性或不稳定性。

⊖ 下面 5 个要点引用了博客 http://webdataanalysis.net/ 中的内容，在此表示感谢。

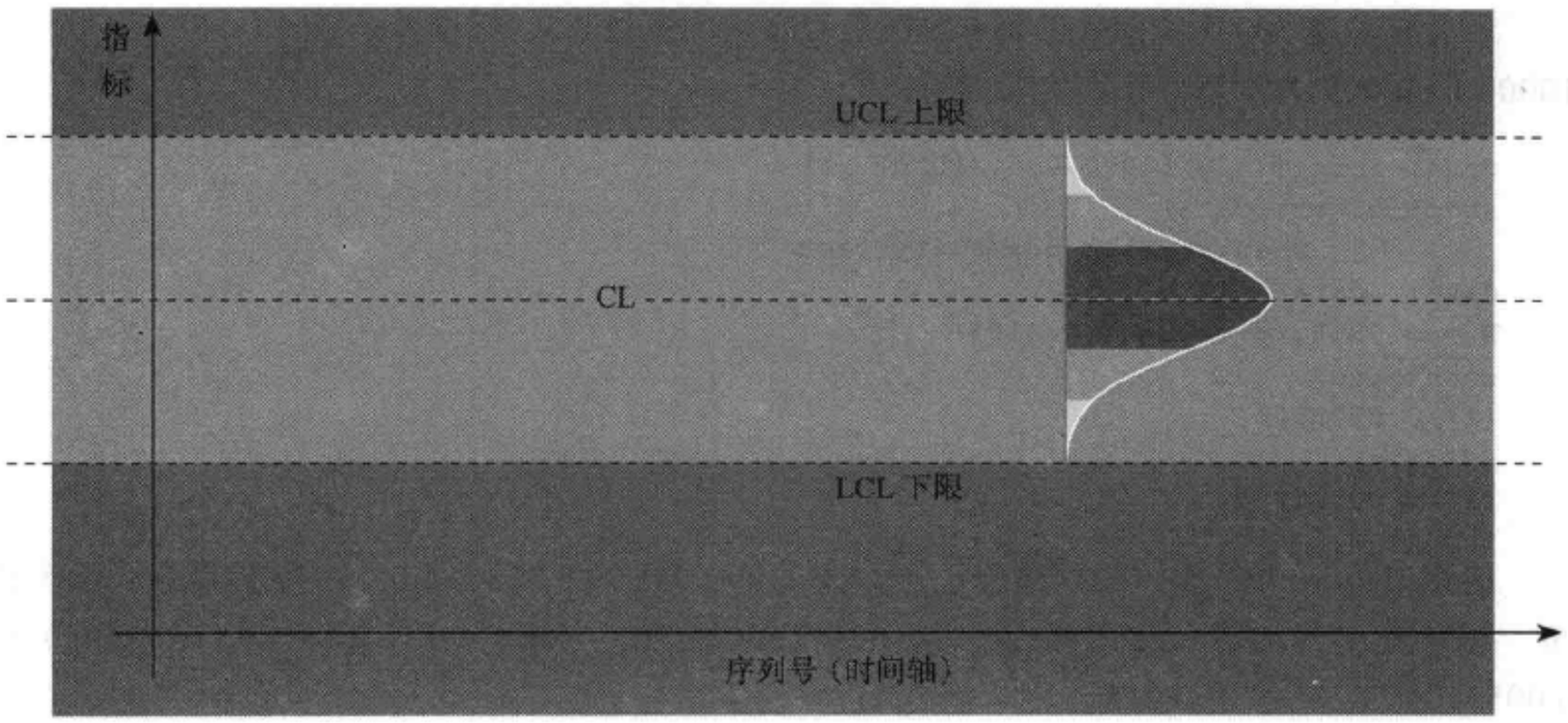


图 4-15 示例图

4.2.2 P 控制图：监控转化率型指标

1. 原理

（1）P 控制图原理

根据中心极限定理规律，当二项分布的样本容量足够大时，分布趋向于均值为$\bar{P}$，标准差为$\sqrt{\frac{\bar{P}(1-\bar{P})}{n}}$的正态分布。下面举例来说明该原理，并使用 R 语言进行模拟，以验证其正确性。

投硬币得到正面或反面的结果是一个典型的二项分布，$P\{A=$正面$\}=0.5$，$P\{A=$反面$\}=0.5$。这里模拟一组实验：投 10000 次（满足样本容量足够大这个条件）硬币，然后统计其正面的数目为：

```
rbinom(1,10000,0.5)
[1]  4994
```

通过 10000 次投币计算 $P\{A=$正面$\}$：

```
rbinom(1,10000,0.5)/10000
[1]  0.4994
```

如果做 500 组这样的实验，则得到 500 个 $P\{A=$正面$\}$：

```
data<-rbinom(500,10000,0.5)/10000
[1] 0.4939 0.5023 0.5012 0.5050 0.4963 0.4950 0.5026 0.4984 0.5055 0.4895
                         ...    ...    ...
[491] 0.5027 0.4932 0.4983 0.5065 0.5028 0.4974 0.4914 0.4961 0.4900 0.4889
```

最后检验这500个$P\{A=$正面$\}$数据是否服从均值是0.5、标准差是sqrt（0.5*（1−0.5）/10000）=0.005的正态分布。代码如下：

```
data<-rbinom(500,10000,0.5)/10000
shapiro.test(data)
        Shapiro-Wilk normality test
data:  data
W = 0.9968, p-value = 0.4185
mean(data)
[1] 0.5002506
sd(data)
[1] 0.005310048
```

使用shapiro.test函数得到检验正态分布的P-value为0.4185>0.05，即判断其服从正态分布。再计算出其均值mean（data），该值是0.5002506，约等于0.5，其标准差sd（data）是0.005310048，约等于0.005。

（2）计算上下限

下面对一个二元取值的随机变量A做如下多组实验：

实验组别	实验次数	A取1的次数	计算A取1的概率
1	$n1$	$n1*p1$	$p1$
2	$n2$	$n2*p2$	$p2$
……			
k	nk	$nk*pk$	pk
……			

当使用3倍标准差下的P控制图来监控A取1时的概率$P\{A=1\}$这个指标时，其公式为：

$$CL=\overline{p}=\frac{\sum p_k n_k}{\sum n_k}$$

$$UCL_k=\overline{p}+3\times\sqrt{\frac{\overline{p}(1-\overline{p})}{n_k}}$$

$$LCL_k=\overline{p}-3\times\sqrt{\frac{\overline{p}(1-\overline{p})}{n_k}} \quad (4\text{-}1)$$

注意：

实际上，不同样本的UCL和LCL是随nk而变化的。

有时可以使用nk的平均值来近似地代替所有的nk，这样一来，所有样本的UCL和LCL等值，也就组成了一条直线。

当使用2倍标准差来区分异常点时，可以把式4-1中的3换成2。

注意P控制图监控数据指标的前提如下：

- 监控转化率型指标：即在大样本（nk）下，二项分布随机变量 A（用户是否到达目标页面）取“到达目标页面”时的概率 $P\{A=$ 到达目标页面 $\}$。
- 数据中每组实验的 nk 要足够大：这是符合中心极限定理规律的前提条件。

2. qcc 函数

qcc 包是专业绘制质量监控图的算法包，其核心是 qcc 函数。该函数的基础形式如下：

```
qcc(data, type, nsigmas = 3, plot = TRUE, ...)
```

其核心参数如表 4-1 所示。

表 4-1 qcc 函数的参数

参数	说明
data	样本数据
size	type="p", "np" 和 "u" 时需要设置
type	绘制控制图的类型如下[⊖]： "xbar"： Xbar 图（均值控制图） "R"： Xbar-R 图（均值 – 极差控制图）；也可以绘制 X-MR 图（单值 – 移动极差控制图） "S"： Xbar-S 图（均值 – 标准差控制图） "xbar.one"： 单值 – 均值控制图 "p"： P 图（用于可变样本量的目标页面到达率） "np"： np 图（用于固定样本量的目标页面到达量）
nsigmas	设置用于计算异常点的上（UCL）下（LCL）限，默认是 3 倍标准差（也叫 3 倍西格玛）
plot	如果为 TRUE（默认情况下），则结果不绘制质量控制图 如果为 FALSE，则结果绘制质量控制图

在绘制 P 控制图时要把 type 参数设置为 "p"。

3. 实例：网站转化率的 P 控制图

下面介绍网站转化率指标。假设把用户是否达到目标页面作为一个随机变量 A，其取值只能为达到（取 1）或未达到（取 0）。在网站无明显优化和改版的情况下，可以近似地把它视为一个服从二项分布的随机变量。具体数据如下：

数据　时间	独立访问用户数（view）	达到目标页面用户数（targit）	转化率（targit/view）
2 月 1 日	10231	201	0.01964617
2 月 2 日	12874	229	0.01778779

⊖ 还可取 u 和 c，但网站分析中使用得比较少。

（续）

数据 时间	独立访问用户数（view）	达到目标页面用户数（targit）	转化率（targit/view）
2月3日	11229	231	0.02057173
========== 中间数据省略展示 ==========			
2月14日	13815	276	0.01997828
2月15日	15766	248	0.01573005

实例的目的是绘制近15日随机变量 A（用户是否达到目标页面）等于1时（能到达目标页面）的概率 $P\{A=1\}$ 的质量控制图，即转化率（targit/view）的质量控制图。

（1）gcc包实现

使用gcc包的gcc函数绘制P控制图。代码如下：

```
library(qcc)
view<-c(10231,12874,11229,9870,11804,11652,13259,11891,12876,14562,12933,13548,
15230,13815,15766)
targit<-c(201,229,231,201,237,224,236,167,213,240,259,241,256,276,248)
sol<-qcc(targit,sizes=view,type=" p" );summary(sol)
Call:
qcc(data = targit, type = "p", sizes = view)
p chart for targit

====== 中间结果省略展示 ======

Number of groups:  15
Center of group statistics:  0.01805889
Standard deviation:  0.1331644
Control limits:
          LCL          UCL
   0.01410932   0.02200847
   0.01453800   0.02157978
          ...          ...
   0.01487727   0.02124051
```

在上述代码中，sol<-qcc（targit,sizes=view,type="p"）为核心语句，它绘制转化率targit/view这个比例型变量的P控制图，并计算相关数据。

另外，summary（sol）语句可以大致查看qcc的处理结果。其中：

```
Center of group statistics:  0.01805889
Standard deviation:  0.1331644
```

表明转化率的均值是0.01805889，标准差是0.1331644 。这两个数据在绘制出的图形下方也有展示。使用sol$center和sol$std.dev命令也可以直接返回该数据。

下面的语句：

```
Control limits:
          LCL          UCL
```

```
0.01410932    0.02200847
0.01453800    0.02157978
       ...    ...    ...
0.01487727    0.02124051
```

表明 3 倍标准差时（nsigma 参数为 3），依据每天转化率计算的 15 个上限 UCL 和下限 LCL 的数据。使用 sol$limits 语句可以直接返回上面的数据。

绘制的结果如图 4-16 所示。图 4-16 中的 Number beyond limits=1 的含义是有 1 天转化率异常。图 4-16 中的第 8 个点为红色，并处于虚线以下，这表明第 8 天的转化率异常。使用 sol$violations[[1]] 命令可以返回检查出的所有异常点序列号向量。

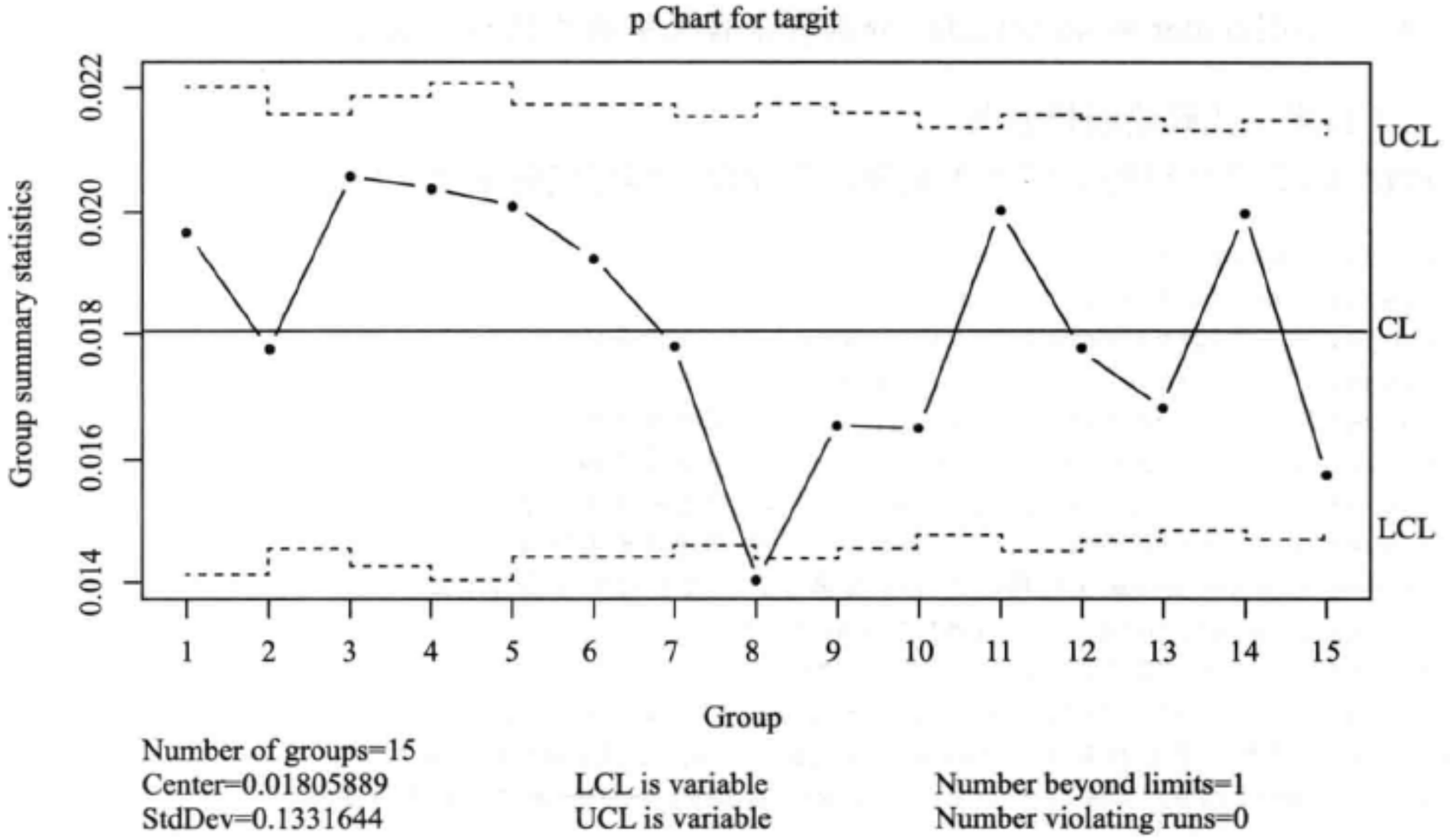


图 4-16 P 控制图

技巧

很多算法的结果对象的结构都很复杂，包含的属性也很多，部分初学者不知如何直接读取或引用其中某一部分的数据。这里介绍一种方法来帮助解决这个问题。以上面的 qcc 函数为例。

如果是通过 sol<-qcc(…) 命令查看的，则 sol 是 qcc 函数的结果对象。

如果使用 str 语句来查看，则结果对象包括的所有属性名称，键入 str（sol）后展示为：

```
List of 11
 $ call      : language qcc(data = targit, type = "p", sizes = view)
 $ type      : chr "p"
 $ data.name : chr "targit"
 $ data      : num [1:15, 1] 201 229 231 201 237 224 236 167 213 240 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ Group  : chr [1:15] "1" "2" "3" "4" ...
```

```
 .. ..$ Samples: NULL
$ statistics: Named num [1:15] 0.0196 0.0178 0.0206 0.0204 0.0201 ...
 ..- attr(*, "names")= chr [1:15] "1" "2" "3" "4" ...
$ sizes     : num [1:15] 10231 12874 11229 9870 11804 ...
$ center    : num 0.0181
$ std.dev   : num 0.133
$ nsigmas   : num 3
$ limits    : num [1:15, 1:2] 0.0141 0.0145 0.0143 0.014 0.0144 ...
 ..- attr(*, "dimnames")=List of 2
 .. ..$ : chr [1:15] "" "" "" "" ...
 .. ..$ : chr [1:2] "LCL" "UCL"
$ violations:List of 2
 ..$ beyond.limits : int 8
```

注意，所有 $ 后的名称，均可以使用 sol$< 名称 > 来读取其 qcc 处理结果对象的属性数据。例如，sol$center 和 sol$std.dev 分别表示转化率的均值和标准差。

（2）自定义代码绘制控制图

依据 4.2.2 节介绍的 P 控制图原理，下面自己编写代码来展示控制图。

```
conver<-targit/view
conver.cl<-sum(targit)/sum(view)
range.3<-3*sqrt(conver.cl*(1-conver.cl)/view)
range.2<-2*sqrt(conver.cl*(1-conver.cl)/view)
conver.ucl.3<-conver.cl+range.3     #3 倍标准差的上限
conver.lcl.3<-conver.cl-range.3     #3 倍标准差的下限
conver.ucl.2<-conver.cl+range.2     #2 倍标准差的上限
conver.lcl.2<-conver.cl-range.2     #2 倍标准差的下限
```

①依据公式 4-1 计算 CL 以及 2 倍和 3 倍标准差下 UCL LCL 的数值

```
col=c("black","green","red","orange")
x.text<-paste("2.",c(1:15),sep="")
plot(conver,type="b",xaxt="n",col=col[1],xlim=c(0,17),ylim=c(0.013,0.023),pch=1
9,main=list(" 转化率 p 控制图 ",cex=2),xlab=" 时间 ",ylab=" 转换率 ")
axis(1,at=1:15,labels=x.text,tick=FALSE);
```

②绘制主体图形

```
lines(seq(0.5,16,1),c(conver.ucl.3[1],conver.ucl.3),type="S",col=col[3])
lines(seq(0.5,16,1),c(conver.lcl.3[1],conver.lcl.3),type="S",col=col[3])
lines(seq(0.5,16,1),c(conver.ucl.2[1],conver.ucl.2),type="S",col=col[4])
lines(seq(0.5,16,1),c(conver.lcl.2[1],conver.lcl.2),type="S",col=col[4])
abline(h=conver.cl,col=col[2])
```

③绘制 2 倍和 3 倍标准差的上下限以及平均值

```
warrings.3<-c(which((conver<conver.ucl.3)==FALSE),
which((conver>conver.lcl.3)==FALSE))   # 处于 3 倍标准差上下限以外的样本点序列号
warrings.2<-c(which((conver<conver.ucl.2)==FALSE),
which((conver>conver.lcl.2)==FALSE))   # 处于 2 倍标准差上下限以外的样本点序列号
points(warrings.2,conver[warrings.2],col=col[4],pch=19)
points(warrings.3,conver[warrings.3],col=col[3],pch=19)
text(warrings.3,conver[warrings.3]," 异常点 ")
```

④把异常点标记颜色和文字

该代码首先依据公式 4-1 公式计算 2 倍和 3 倍标准差下的上限 UCL 和下限 LCL。然后使用 plot 函数绘制出监控指标（转化率）随时间变化的散点图。之后使用红色绘制 3 倍标准差下的上下限横线，使用橘黄色绘制 2 倍标准差下的上下限横线，使用绿色绘制 y 轴等于均

值的水平直线。

最后的绘制结果如图 4-17 所示。

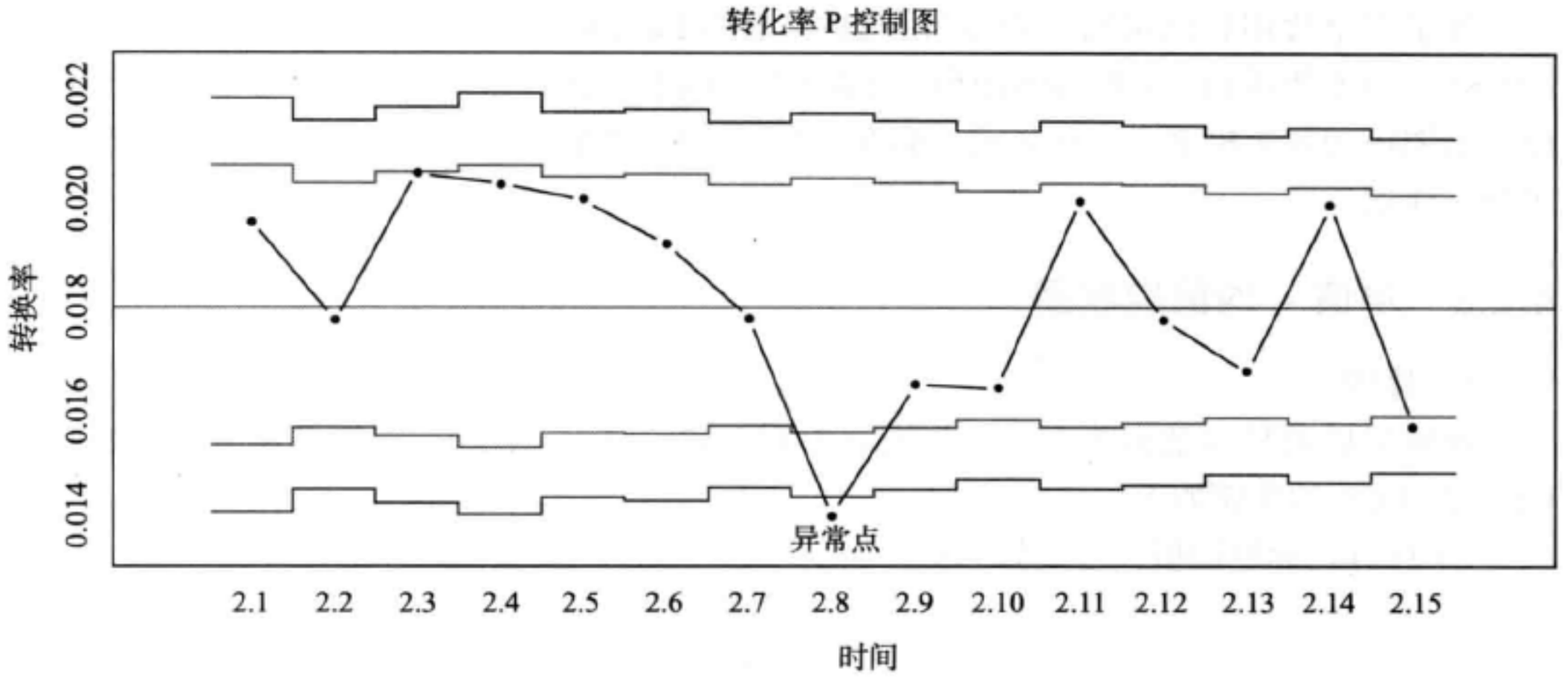


图 4-17 转化率指标的 P 控制图

对于公式 4-1 中的 nk 也可以使用 n 的平均值来近似代替，即把上面代码中的：

```
range.3<-3*sqrt(conver.cl*(1-conver.cl)/view)
range.2<-2*sqrt(conver.cl*(1-conver.cl)/view)
```

变为：

```
range.3<-3*sqrt(conver.cl*(1-conver.cl)/mean(view))
range.2<-2*sqrt(conver.cl*(1-conver.cl)/mean(view))
```

并修改其他配套代码，这里就不给出详细代码了，读者可以自己尝试编写。最后可以绘制出如图 4-18 所示的 P 控制图。

图 4-18 转化率指标的 P 控制图

图 4-18 同图 4-17 最大的不同是，红色和橘黄色的 UCL 和 LCL 横线不再是曲线了，而是直线。

除了完全使用自己编写的命令外，也可以使用 qcc 函数（此时通常把 plot 参数设置为 FALSE，即不使用 qcc 函数绘制图形）计算 CL、UCL、LCL 异常点序列号等数据，然后再编写绘图语句展示出来。这种方式，会在 4.2.3 节中以单值均值控制图的绘制为例讲解，这里暂不详述。

4.2.3 单值 – 均值控制图

1. 原理

如果指标的样本数据为 k 个，并记录为 $x1$，$x2$，$x3$，…，xk，则计算单值均值控制图 CL UCL LCL 的步骤如下：

1）计算指标平均值，公式如下：

$$\overline{x}=\frac{\sum x_i}{k}$$

2）计算样本的移动极差，公式如下：

$$\mathrm{MR}=|X_i-X_{i-1}|$$

3）计算移动极差均值（k 个样本数据产生 $k-1$ 个移动极差），公式如下：

$$\overline{MR}=\frac{\sum MR_i}{k-1}$$

4）计算 CL，公式如下：

$$CL=\overline{x}$$

5）计算 UCL 和 LCL（在 3 倍标准差情况下）。

$$UCL=\overline{x}+3\times\frac{\overline{MR}}{d_2}$$

$$UCL=\overline{x}-3\times\frac{\overline{MR}}{d_2}$$

其中 d_2 等于 1.128。如果在 2 倍标准差情况下，则公式中的 3 变为 2。

2. 实例：网站客单价的单值均值控制图

以下是某网站 3 月份半个月的客单价统计数据以及计算 UCL 和 LCL 过程。

数据	日期	客单价（元）	客单价移动极差
	2013.3.1	103.76	NA
	2013.3.2	129.12	25.36
	2013.3.3	107.30	21.82
	…	…	…
	2013.3.15	82.76	2.47
	客单价均值 :103.4793		

（续）

数据	日期	客单价（元）	客单价移动极差
	移动极差均值：11.32429		
	3 倍标准差下，UCL=103.4793+3* 11.32429/1.128=133.5971		
		LCL=103.4793-3* 11.32429/1.128=73.3615	

（1）qcc 包实现

把 qcc 包中的核心函数 qcc 的参数 type 设置为 "xbar.one"，即可绘制和计算单值均值控制图。代码如下：

```
price<-c(103.76,129.12,107.30,97.45,105.10,115.78,105.21,98.78,101.74,96.53,97.
99,114.20,116.18,80.29,82.76)
    sol<-qcc(price,type="xbar.one");
```

绘制结果如图 4-19 所示。

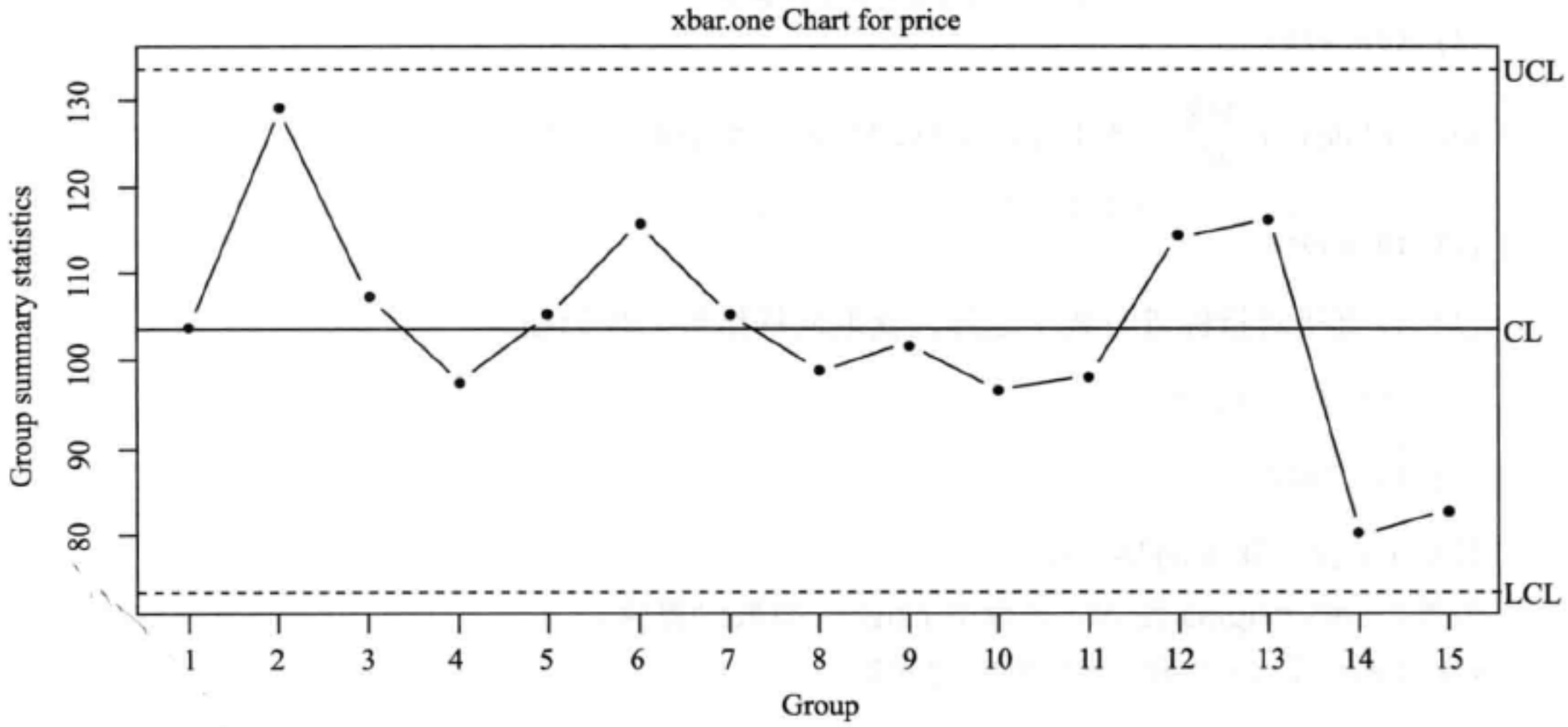


图 4-19 单值均值控制图

对于 sol 内部的数据，首先通过 str（sol）命令查看其属性概要。代码如下：

```
str(sol)
List of 11
 $ call      : language qcc(data = price, type = "xbar.one")
 $ type      : chr "xbar.one"
 $ data.name : chr "price"
 $ data      : num [1:15, 1] 103.8 129.1 107.3 97.5 105.1 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ Group  : chr [1:15] "1" "2" "3" "4" ...
```

```
 .. ..$ Samples: NULL
$ statistics: Named num [1:15] 103.8 129.1 107.3 97.5 105.1 ...
 ..- attr(*, "names")= chr [1:15] "1" "2" "3" "4" ...
$ sizes     : int [1:15] 1 1 1 1 1 1 1 1 1 1 ...
$ center    : num 103
$ std.dev   : num 10
$ nsigmas   : num 3
$ limits    : num [1, 1:2] 73.4 133.6
 ..- attr(*, "dimnames")=List of 2
 .. ..$ : chr ""
 .. ..$ : chr [1:2] "LCL" "UCL"
$ violations:List of 2
 ..$ beyond.limits : int(0)
 ..$ violating.runs: num(0)
 - attr(*, "class")= chr "qcc"
```

在上述代码中，sol$ center 是 CL，即客单价指标的均值 mean（price）。

```
sol$ center          # 直接读取 qcc 函数产生的对象
[1] 103.4793
```

sol$ std.dev 是$\frac{MR}{d_2}$，即指标样本移动极差的均值除以 1.128。

```
sol$ std.dev         # 直接读取 qcc 函数产生的对象
[1] 10.03926
```

也可以直接对指标进行数学运算，获取同样结果。代码如下：

```
mr<-abs(diff(price))
mean(mr)/1.128
[1] 10.03926
```

其中 mr 就是样本的移动极差。

另外，sol$ nsigma 是 qcc 函数的 nsigma 参数设置值。

sol$ limits 是 LCL 和 UCL 组成的向量。

```
sol$ limits
       LCL      UCL
  73.36155  133.5971
```

依据 LCL 和 UCL 的定义：sol$ center + nsigma*sol$ std.dev，也可以直接对指标进行数学运算，获取同样结果。代码如下：

```
cl<-mean(price)
mr<-abs(diff(price))
sd.dev<- mean(mr)/1.128
nsigma<-3
lcl<-cl-nsigma* sd.dev;lcl
[1] 73.36155
ucl<-cl+nsigma* sd.dev;ucl
[1] 133.5971
```

sol$violations$beyond.limits 是异常点的序列号。上例中并无异常点，这里调整 qcc 的 nsigma 参数为 1.5，使其计算在 1.5 倍标准差下的控制图，然后查看其异常点信息。代码如下：

```
sol<-qcc(price,type="xbar.one",nsigma=1.5,plot=FALSE)
sol$violations$beyond.limits
[1]  2 14 15
```

可知，在 1.5 倍标准差下，使用单值均值控制图时，指标的第 2、14、15 这 3 个点均为异常点。

（2）编写代码绘制控制图

在 4.2.2 节中，通过不使用 qcc 函数的方式绘制了 p 控制图，这里取一个折中的方式：通过 qcc 函数计算单值均值控制图的核心数据（包括 CL、UCL、LCL、异常点序列号等），并自己编写绘图命令加以展示。

```
    sol.3<-qcc(price,type="xbar.one",plot=F)
    sol.2<-qcc(price,type="xbar.one",plot=F,nsigma=2)

    col=c("black","green","red","orange")
    x.text<-paste("3.",c(1:15),sep="")
    plot(price,type="b",xaxt="n",col=col[1],xlim=c(0,17),ylim=c(min(sol.3$limits[1],
min(price))-10,max(sol.3$limits[2],max(price))+10),
         pch=19,main=list("日客单价：单值-均值质量控制图",cex=1.5),xlab="时间",ylab="
日客单价（元）")
    warrings.3<-sol.3$violations$beyond.limits
    warrings.2<-sol.2$violations$beyond.limits
    axis(1,at=1:15,labels=x.text,tick=FALSE);

    abline(h=sol.3$limits[2],col=col[3])
    abline(h=sol.3$limits[1],col=col[3])
    abline(h=sol.2$limits[2],col=col[4])
    abline(h=sol.2$limits[1],col=col[4])
    abline(h=sol.3$center,col=col[2])

    points(warrings.2,price[warrings.2],col=col[4],pch=19)
    points(warrings.3,price[warrings.3],col=col[3],pch=19)
    text(warrings.3,price[warrings.3],"异常点")
```

通过 qcc 函数可得到 2 倍和 3 倍标准差下返回的对象 sol.2 和 sol.3。注意，qcc 函数中的 plot 设置为 FALSE，即不通过 qcc 函数绘制图形。

然后逐一读取核心数据（以 3 倍标准差为例）。

- CL：sol.3$center
- UCL：sol.3$limits[2]
- LCL：sol.3$limits[1]
- 异常点序列号向量：sol.3$violations$beyond.limits

相关的绘图函数与 4.2.2 节实例中的命令基本相同：绘制了 CL（绿色水平线）、2 倍标准

差下 UCL/LCL（橘色水平线）、3 倍标准差下 UCL/LCL（红色水平线），并把处于 3 倍标准差外的样本点标记为“异常点”。绘制结果如图 4-20 所示。

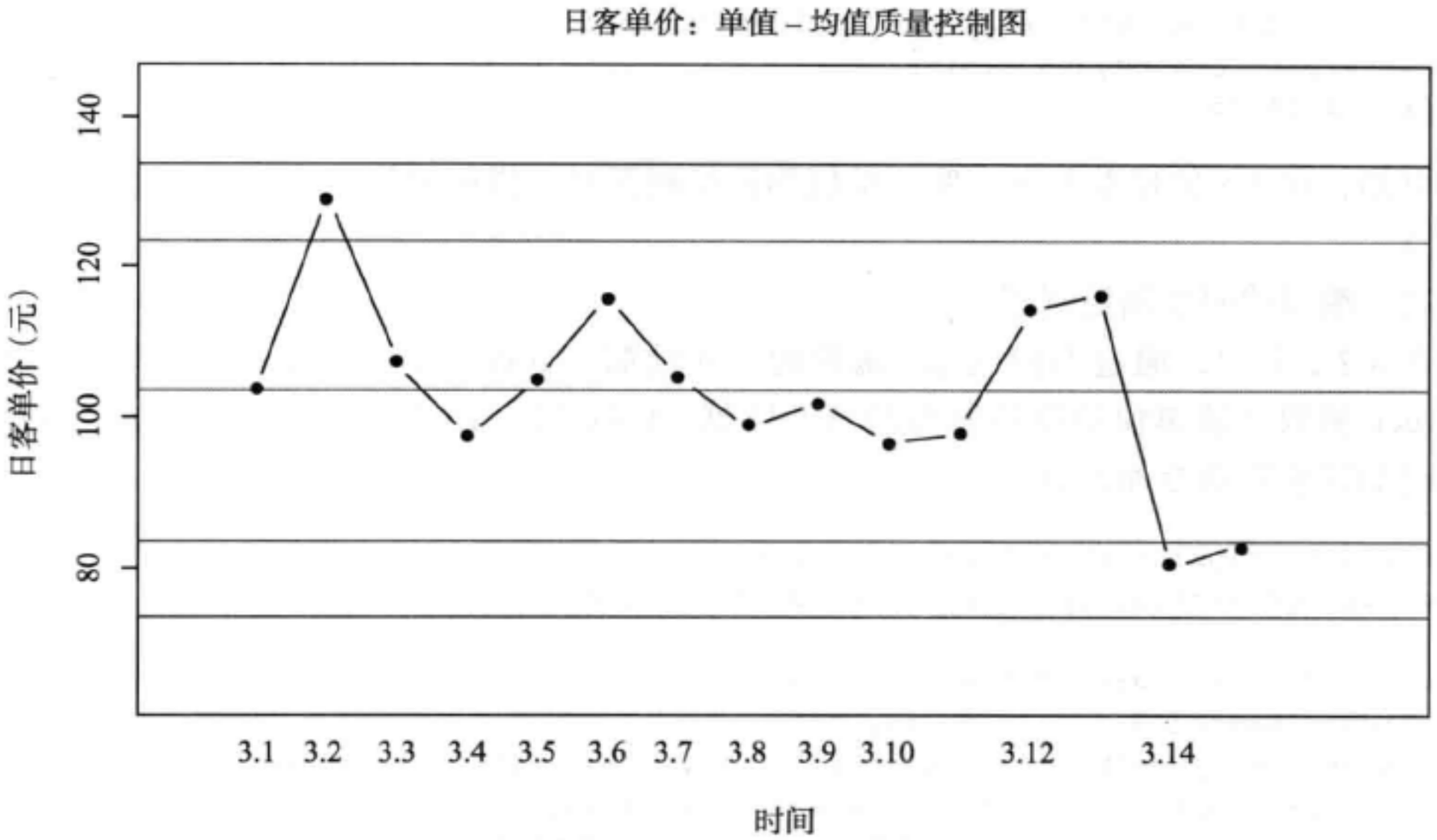


图 4-20　客单价指标的单值均值控制图

3. 扩展：均值控制图

上面讲解的是单值 – 均值控制图，在原始数据中每个样本内只有 1 个数据（所以称之为“单值”）。如果每个样本内有 3 个数据（不再是单值了），即一共使用 3 组类似于上例的数据来研究客单价指标，那么使用什么来处理呢？可以使用均值控制图（可以理解为“多值” – 均值控制图）来处理。

假设有如表 4-2 所示的数据：共有 k 个样本，每个样本内有 3 个数值。

表 4-2　样本的均值极差示例

样本序列号	每个样本的 3 组组号			每个样本的均值 $\bar{X}$	每个样本的极差 R
	$X1$	$X2$	$X3$		
1	$x11$	$x12$	$x13$	mean（$x11, x12, x13$）	max（$x11, x12, x13$）– min（$x11, x12, x13$）
2	$x21$	$x22$	$x23$	mean（$x21, x22, x23$）	max（$x21, x22, x23$）– min（$x21, x22, x23$）
3	$x31$	$x32$	$x33$	mean（$x31, x32, x33$）	max（$x31, x32, x33$）– min（$x31, x32, x33$）
.....					
k	$xk1$	$xk2$	$xk3$	mean（$xk1, xk2, xk3$）	max（$xk1, xk2, xk3$）– min（$xk1, xk2, xk3$）

在 3 倍标准差下：

$$\mathrm{CL}=\overline{\overline{X}}$$

$$\mathrm{UCL}=\overline{\overline{X}}+A_2\,\overline{R}$$

$$\mathrm{LCL}=\overline{\overline{X}}-A_2\,\overline{R}$$

其中$\overline{\overline{X}}$是每个样本的均值的均值，$\overline{R}$是每个样本的极差均值。A_2是随样本内组数而变的常数。在上例中，每个样本内有3组数值，A_2=1.023，如果是2组，则A_2=1.88，如果是4组，则A_2=0.729。

对于其他情况下的A_2取值，读者可以查阅相关文献资料，这里不再详述。

使用qcc函数，并设置type参数为"xbar"，可以绘制和计算均值控制图。对于上例，可以把上述数据存储于k行3列的数据框或矩阵data对象中，然后使用qcc（data,type="xbar"）绘制在3倍标准差下的均值控制图。同时，通过设置nsigma参数也可以绘制和计算不同倍数标准差下的均值控制图。

4.2.4 单值－移动极差控制图

单值－移动极差控制图与单值－均值控制图的使用场景类似，一般情况下可以把这两种图都绘制出来，综合监控数据的波动和异常。

1. 原理

如果指标的样本数据为k个，并记录为$x1$，$x2$，$x3$，…，xk，则计算单值－移动极差控制图CL UCL LCL的步骤如下：

1）计算指标平均值，公式如下：

$$\overline{x}=\frac{\sum x_i}{k}$$

2）计算样本的移动极差，公式如下：

$$\mathrm{MR}=|X_i-X_{i-1}|$$

3）计算移动极差均值（k个样本数据产生$k-1$个移动极差），公式如下：

$$\overline{MR}=\frac{\sum MR_i}{k-1}$$

4）计算CL，公式如下：

$$CL=\overline{MR}$$

5）计算UCL和LCL（在3倍标准差情况下），公式如下：

$$UCL=\overline{MR}+3\times\overline{MR}\times 0.7557748$$

$$UCL=0$$

如果是在2倍标准差的情况下，则公式中的3变为2。

2. 实例：网站客单价的单值－移动极差控制图

仍使用4.2.3节实例中客单价的数据为例，绘制单值－移动极差控制图。

（1）qcc包实现

和单值－均值控制图不同，单值－移动极差控制图在qcc包中没有直接对应的type参数值，只能先对原始数据进行初级处理，然后使用用于绘制均值－极差控制图的方式（type参数取“R”）进行绘制。代码如下：

```
sol<-qcc(data.frame(price[-length(price)],c(price[-1])),type="R",nsigma=2)
```

注意，不能直接使用 price 作为输入，要先对其进行初步处理，并组成一个 2 列的数据框 data.frame（price[-length（price）],c（price[-1]））。其中，首列是删除 price 中最后一个元素的其余数据，第二列是删除 price 中第一个元素的其余数据。并且通过设置 nsigma 为 2，增强了监控门限。

绘制的图形如图 4-21 所示。

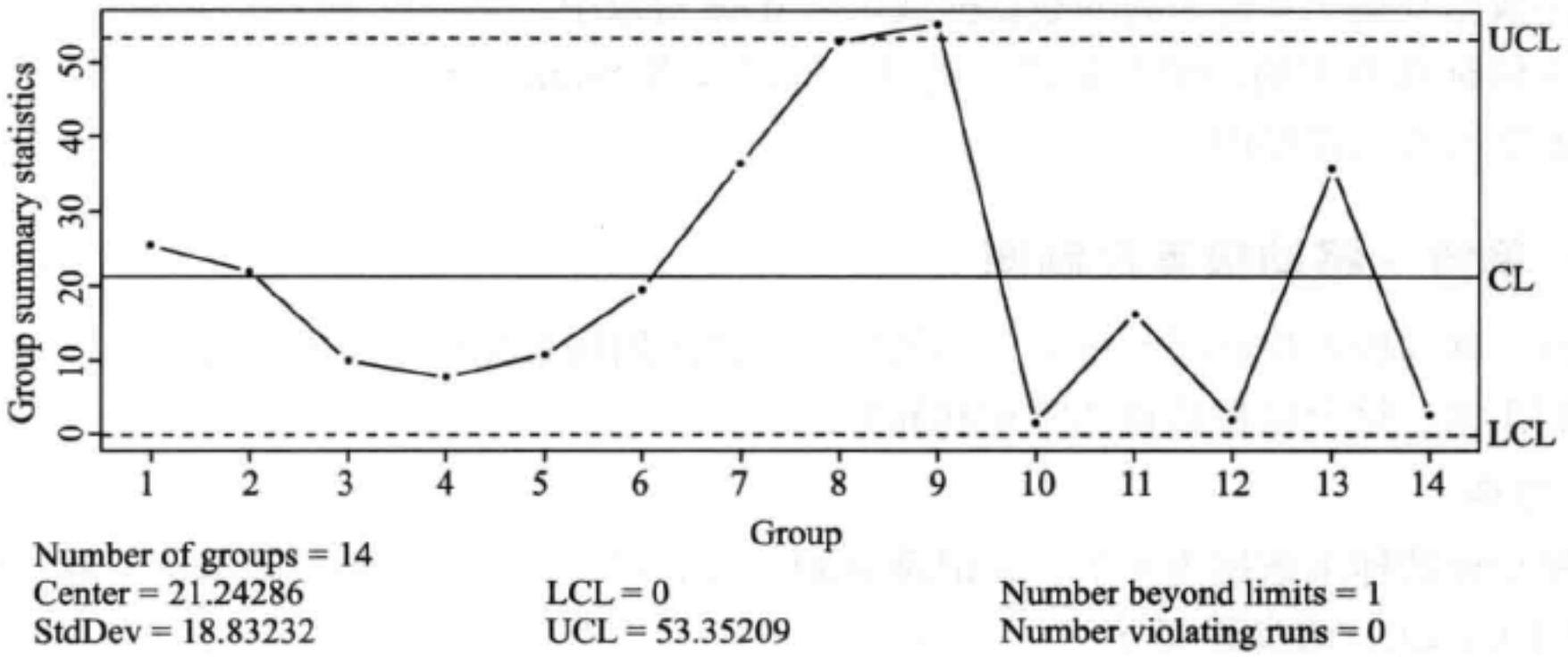


图 4-21 单值 – 移动极差控制图

sol 内部的属性数据，可以参考上面两个类型控制图的讲解。这里只补充介绍 CUCL 以及 LCL 的其他计算方式。

对于均值 – 移动极差控制图的 LCL 和 UCL，除了可以读取 qcc 函数产生的结果 sol$ limits 外，也可依据公式对其进行数学运算，以获取同样结果。代码如下：

```
mr<-abs(diff(price))
cl<-mean(mr)
nsigma=2
ucl<-cl+nsigma*0.7557748*cl;
lcl<-0;
```

（2）编写代码绘制控制图

同 4.2.3 节的实例一样，这里要使用 qcc 计算单值 – 移动极差控制图的核心数据，然后编写绘图命令来展示。代码如下：

```
sol.3<-qcc(data.frame(price[-length(price)],c(price[-1])),type="R",nsigma=3,plot=F)
sol.2<-qcc(data.frame(price[-length(price)],c(price[-1])),type="R",nsigma=2,plot=F)
mr<-abs(diff(price))

col=c("black","green","red","orange")
x.text<-paste("3.",c(2:15),sep="")
plot(mr,type="b",xaxt="n",col=col[1],
     xlim=c(0,17),ylim=c(0,max(sol.3$limits[2],max(mr))+10),
     pch=19,main=list("日客单价：单值 - 移动极差质量控制图",cex=1.5),
     xlab="时间",ylab="日客单价（元）")
```

```
warrings.3<-sol.3$violations$beyond.limits
warrings.2<-sol.2$violations$beyond.limits
axis(1,at=1:14,labels=x.text,tick=FALSE);

abline(h=sol.3$limits[2],col=col[3])
abline(h=sol.3$limits[1],col=col[3])
abline(h=sol.2$limits[2],col=col[4])
abline(h=sol.2$limits[1],col=col[4])
abline(h=mean(mr),col=col[2])

points(warrings.2,mr[warrings.2],col=col[4],pch=19)
points(warrings.3,mr[warrings.3],col=col[3],pch=19)
text(warrings.3,mr[warrings.3],"异常点")
```

上面的代码与4.2.3节实例的代码基本相同：绘制了CL（绿色水平线）、2倍标准差下UCL/LCL（橘色水平线）和3倍标准差下UCL/LCL（红色水平线），并把处于3倍标准差外的样本点标记为“异常点”。其不同之处主要是：

- 核心函数qcc的type参数设置为"R"，并对price进行初级处理，使用一个2列的数据框作为输入。
- 对于后续的绘图操作，其主体数据均不再使用price，而是使用price样板的移动极差mr。

绘制的控制图如图4-22所示。

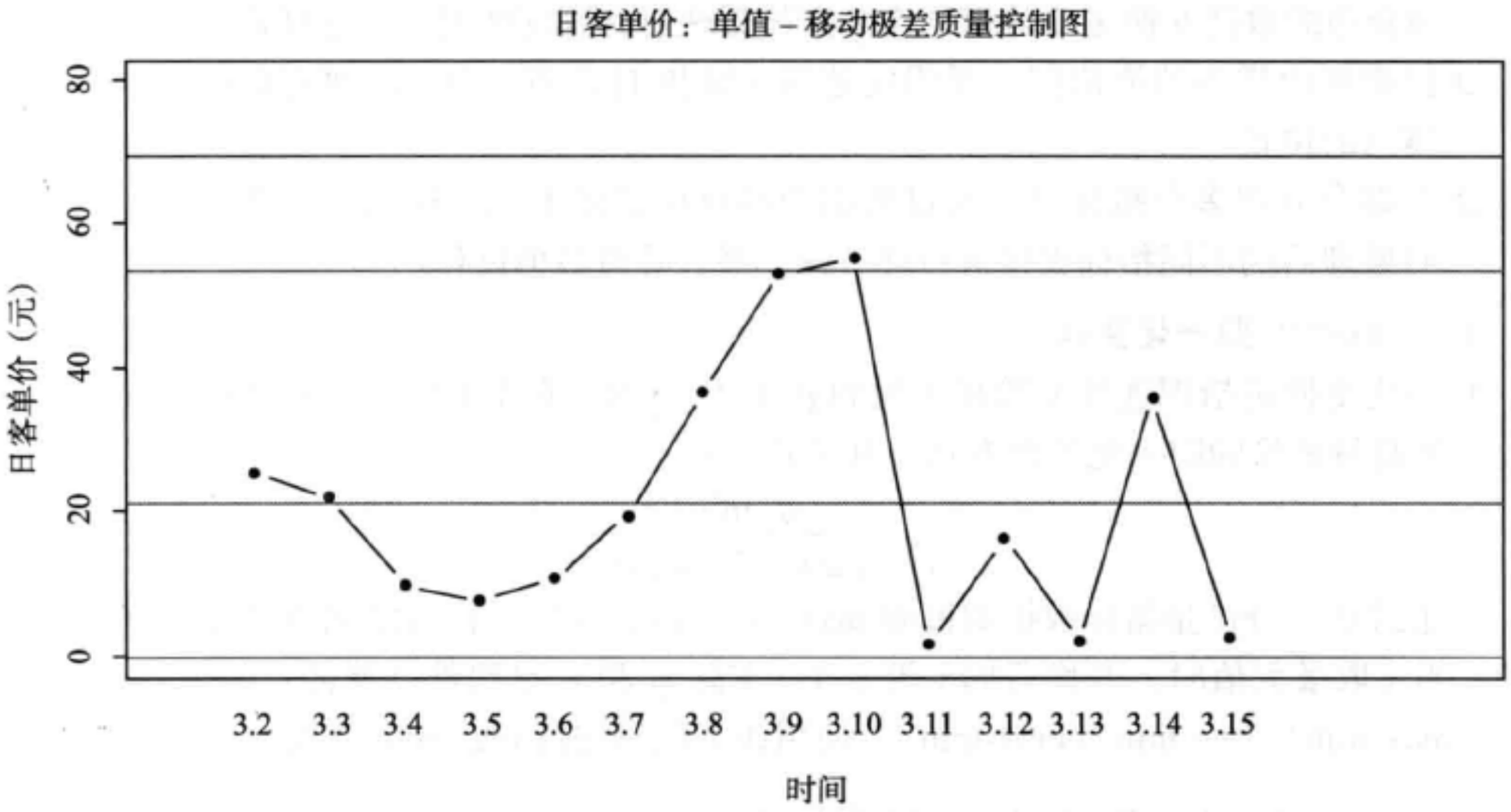


图4-22 客单价指标的单值－移动极差控制图

3. 扩展：均值－极差控制图

如果每个样本内有3个数据（不再是单值了），即一共使用3组类似于上例的数据来研

究客单价指标（和 4.2.3 节类似），就可以使用均值 – 极差控制图来处理，其原理是通过计算每个样本的极差后监控指标样本来实现。

在 3 倍标准差下，使用如下公式：

$$\mathrm{CL}=\overline{R}$$

$$\mathrm{UCL}=D_4\,\overline{R}$$

$$\mathrm{LCL}=D_3\,\overline{R}=0$$

其中$\overline{R}$是每个样本极差的均值。D_4 是随样本内组数而变的常数。在上例中，每个样本内有 2 组数值，则 D_4=3.267；如果有 3 组数值，则 D_4=2.575，如果有 4 组数值，则 D_4=2.282。D_3 则始终是 0。

均值 – 极差控制图的处理方式也是：使用 qcc 函数，并设置 type 参数为 "R"。其原始数据的格式同均值控制图一样，这里就不详细介绍了。

4.3 连续型指标的对比

指标对比是数据分析最常用的方法，因为静态地看一个指标数据无法得到有价值的信息，只有与其他数据对比，才能综合分析出有用的结论。

4.3.1 数据变换

下面介绍的数据变换主要是指把连续型数据映射到固定区间内，这样做有以下好处：

- 把指标内样本的取值归一至固定范围（如 [0,1] 范围）内，以便观察该样本在指标整体内的位置。
- 在综合处理多个指标时，通过数据变换化可消除不同指标间度量单位的影响。可以对处理后的不同指标直接进行加、减、乘、除等数值操作。

1. max-min 归一化变换

归一化变换是指把连续型数据映射到 [0,1] 区间内，它主要用于对比指标内样本。max-min 变换是最常见的归一化处理方式，其公式为：

$$yi=\frac{xi-\min(x)}{\max(x)-\min(x)}$$

在上式中，分母是指标数据的极差 max（x）- min（x）。当 x 取最小值时，变换后的 y 等于 0，当 x 取最大值时，变换后的 y 等于 1。实际 yi 和 xi 是线性变换的：$yi=a\times xi+b$，其中 a=1/（max-min），b= - min/（max-min）。使用如下代码进行 max-min 变换。

```
(data-min(data))/(max(data)-min(data))
```

下面分析 15 个商品被收藏的次数指标 item.like。首先对原始数据 item.like 进行 max-min 变换，并产生 y. item.like 对象。代码如下：

```
y. item.like <-(item.like-min(item.like))/(max(item.like)-min(item.like));y.item.like
```

```
 [1] 0.48064714 1.00000000 0.55314356 0.35142331 0.50808929 0.72680729
 [7] 0.51034200 0.37866066 0.43927913 0.33258243 0.36248208 0.69445013
[13] 0.73499898 0.00000000 0.05058366
```

在变换数据的基础上，把它平均分为 5 个等级，其取值依次为 1 ～ 5。代码如下：

```
y. item.like <-round(y. item.like*4+1)
[1] 3 5 3 2 3 4 3 3 3 2 2 4 4 1 1
```

在上面的代码中，先把 y. item.like 乘以 4，使取值范围变为 [0,4]。之后加 1，使取值范围变为 [1,5]。最后对其做四舍五入计算，得到 item.like 指标的 5 分制变换。

2. 标准化变换

标准化变换主要用于去除指标的度量单位影响（去纲量化），这在综合处理不同指标时尤为重要，其公式为：

$$yi = \frac{xi - \min(x)}{\mathrm{sd}(x)}$$

其中，mean(x) 是指标样本的均值，sd(x) 是指标样本的标准差。变换后样本的均值为 0，标准差为 1。在 R 语言中，scale 函数用于数据的标准化变换。

在聚类分析中，由于涉及多个指标间的综合作用，即要对不同指标数据做加、减、乘、除数值计算（主要是计算样本间的距离），因此在计算前需要对不同指标去纲量化，也就是说，必须使用 scale 做标准化变换。

这里以对比 14 个文章内容页面的热度为例，说明标准化变换的应用。下面是 14 篇文章内容页面的页面浏览量（pv）、访问次数（visit）、跳出率（bounce）、下载按钮点击量（download）、投票喜欢（like）、投票不喜欢（unlike）。

数据	pv	visit	bounce	download	like	unlike
1	532251	389721	0.8661	233	2442	1065
2	574540	398998	0.8745	247	2555	915
.....						
14	635077	436527	0.8476	329	3748	3393

首先，对各个指标做标准化变换，以去纲量化。然后把去纲量后的数据相加（其中由于跳出率（bounce）和投票不喜欢（unlike）指标是负指标，即越大越不好，因此要对它们做相减操作）。最后得到一个综合指标：文章热度 item.hot。代码如下：

```
    item.hot<-scale(item$pv)+scale(item$visit)-scale(item$bounce)+scale(item$downlo
ad)+scale(item$like)-scale(item$unlike)
    item.hot<-(item.hot-min(item.hot))/(max(item.hot)-min(item.hot))
```

绘制结果如图 4-23 所示。

以上只是为展示标准化和归一化的应用而举的关于内容页面热度的例子，实际上往往还要实现数据降维（因子分析或主成分分析），以及确定指标间的权重等步骤，这样才能更好

地分析内容页面的热度。

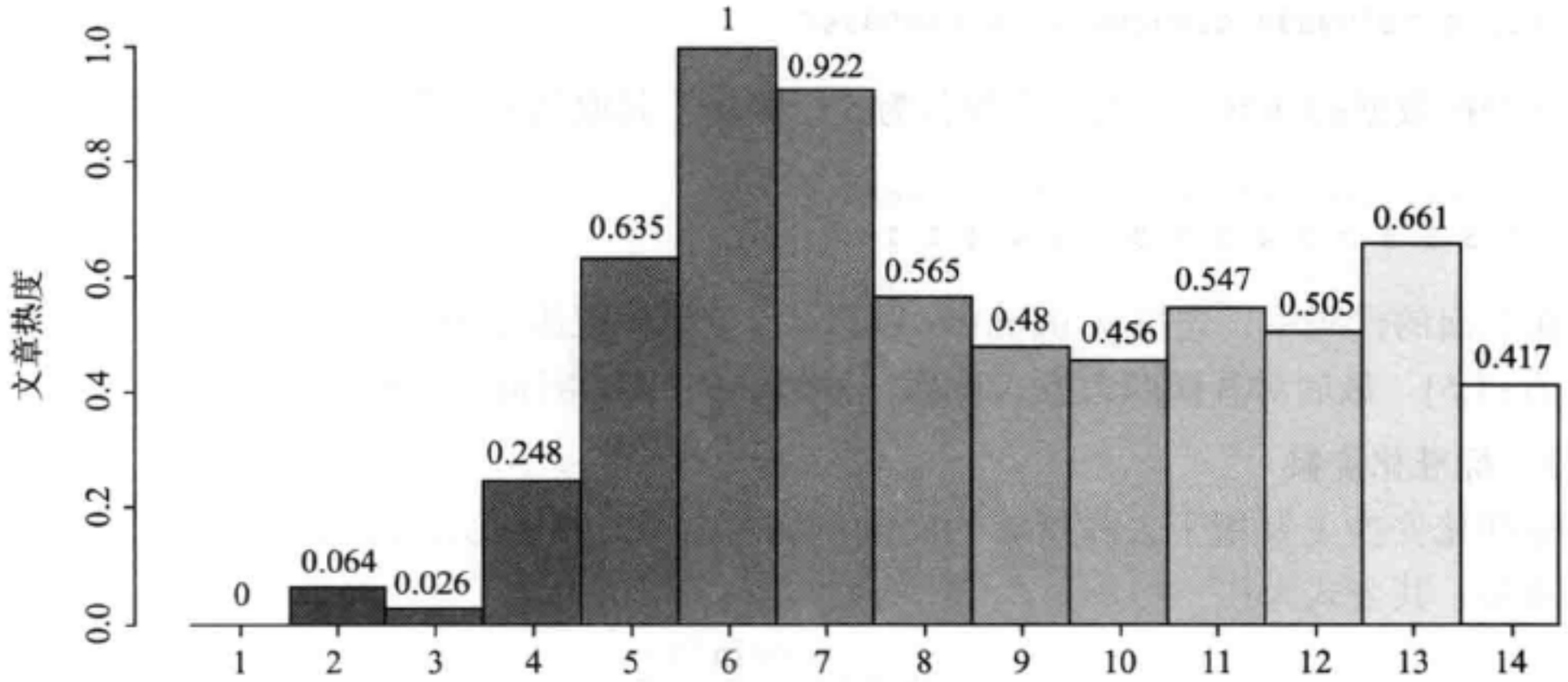


图 4-23 文章热度直方图

4.3.2 假设检验

假设检验是用反证法来证明某一假设是否成立。

1）假定这个假设 H0 是成立的，并构造出一个服从某一种分布 $F(x)$ 的统计量 X（$X \sim F(x)$）。

2）计算服从分布 $F(x)$ 的随机变量 x 大于（或小于）统计量 X 的概率 $P\{X<x\}$（或 $P\{X>x\}$）。

说明

在 R 语言中，$P\{X<x\}$（或 $P\{X>x\}$）常记为 p-value。

3）选定一个显著性水平 alpha，如果 p-value <alpha，则认为原来的假设 H0 不成立。如果 p-value >alpha，则认为原来的假设 H0 成立。不同检验标准下，显著水平 alpha 的典型取值如表 4-3 所示。

表 4-3 显著水平 alpha 的典型取值

对假设的检验标准	显著性水平 alpha 的值
放松的检验标准	0.1 或 0.075
一般情况	0.05
严格的检验标准	0.025 或 0.01

第一类错误：使用上面的假设检验时，虽然判定结果显示 H0 不成立，但实际 H0 是成立的。也可以认为显著性水平 alpha 是发生第一类错误的概率，即 alpha=P{判定结果显示 H0 不成立 | 实际 H0 成立}。可见如果设置较小的 alpha 参数，则由 p-value <alpha 推理出原假设 H0 不成立的结论就越可靠。

第二类错误：使用上面的假设检验时，如果判定结果显示 H0 成立，但实际 H0 是不成立的。alpha 越小，则由 p-value >alpha 推理出原假设 H0 成立的结论越不可靠。

注意

如果要做严格的假设检验，则可以使用如下规则：

如果 p-value <alpha，则认为原来的假设 H0 不成立。

如果 p-value >1-alpha，则认为原来的假设 H0 成立。

如果 alpha <p-value <1-alpha，则不给出任何结论。

而一般情况下只要使用如下规则便可：

如果 p-value <alpha，则认为原来的假设 H0 不成立。

如果 p-value >alpha，则认为原来的假设 H0 成立。

4.3.3 相同指标内的两组数据对比：T 检验

在两种不同的情况下，2 组相同指标间经常会进行对比。例如，经常会对网站进行改版或推广，通常这时候都会检验改版（或推广）前后某一指标是否有明显不同（经常是指其均值不同）。实际这是在检验一个二元分类变量是否对一个连续变量有明显影响。

1. 原理

在不同前提下可以构建的统计量，以及在满足假设 H0 时服从的分布状态如表 4-4 所示。总前提是涉及检验的两组指标均要服从正态分布。

表 4-4 构建统计量

	前提	假设 H0	构建的统计量	服从分布	R 函数
1	已知指标方差 σ^2	$\mu=\mu_0$	$\frac{\overline{X}-\mu_0}{\sigma/\sqrt{n}}$	$N(0,1)$	shapiro.test（x）
2	未知指标方差	$\mu=\mu_0$	$\frac{\overline{X}-\mu_0}{S/\sqrt{n}}$（其中 S 是样本标准差）	$t(n\text{-}1)$	t.test（x）
3	已知两个指标的方差 σ_1^2 和 σ_2^2	$\mu_1=\mu_2$	$\frac{\overline{X}-\overline{Y}}{\sqrt{\frac{\sigma_1^2}{n_1}+\frac{\sigma_2^2}{n_2}}}$	$N(0,1)$	shapiro.test（x）
4	已知两个指标的方差相等但不知具体数值	$\mu_1=\mu_2$	$\frac{\overline{X}-\overline{Y}}{S_\mu\sqrt{\frac{1}{n_1}+\frac{2}{n_2}}}$	$t(n1+n2\text{-}2)$	t.test（x,y）
5	未知两个指标的方差	$\mu_1=\mu_2$	$T=\frac{\overline{X}-\overline{Y}}{\sqrt{\frac{S_1^2}{n_1}+\frac{S_2^2}{n_2}}}$	$t(\hat{v})$ 其中 $\hat{v}$ 参考公式 4-3-1	t.test（x,y）

其中：

$$\hat{v}=\left(\frac{S_1^2}{n_1}+\frac{S_2^2}{n_2}\right)^2\Bigg/\left(\frac{(S_1^2)^2}{n_1^2(n_1-1)}+\frac{(S_2^2)^2}{n_2^2(n_2-1)}\right) \tag{4-2}$$

由于一般情况下不知道指标的真实方差（情况 1 和 3 很难满足），因此会用样本方差来

代替真实方差（比如表4-4中的2、4、5这几种情况）。在R语言中，t.test可以用于判定指标1的均值等于/大于/小于特定值，以及判定指标1均值和指标2均值间的等于/大于/小于关系。

2. t.test函数

t.test函数用于在未知指标方差的情况下，检验两组指标的均值。t.test函数的基本形式如下：

```
t.test (x,y=NULL, alternative=c ("two.sided" , " less" ," greater") , mu=0, paired= FALSE, var.equal=FALSE,…)
```

具体参数说明如表4-5所示。

表4-5 t.test函数的参数

参数	说明
x,y	如果设置了y参数，则t.test用于对x和y两组指标做均值大小检验。 如果不设置y（NULL）参数，则t.test用于检验x均值与定值mu的大小关系
mu	只有不设置y参数时，mu才有意义。具体意义参考x、y参数说明
alternative	不同alternative的取值含义不同，具体如表4-6所示
paired	如果取TRUE，则对具有相同数目样本的两组指标间做对比检验。 如果取FALSE（默认取值），则可对不具有相同数目样本的两组指标间做对比检验
var.equal	如果取TRUE，则在已知x和y具有相同方差的情况下做T检验 如果取FALSE（默认取值），则在x和y不具有相同方差的情况下做T检验

表4-6 alternative参数取值说明

alternative取值	"two.sided"	"less"	"greater"
不设置y参数	检验定值mu==x均值	检验定值mu＜x均值	检验定值mu＞x均值
设置y参数	检验y均值==x均值	检验y均值＜x均值	检验y均值＞x均值

3. 实例：网站推广对页面浏览量指标的显著影响

在对网站投放某一广告后，通常希望获知它对网站流量pv是否有显著的影响。下面监控广告推广后10天的pv数据。代码如下：

```
pv.new<-c(10740,13368,11730,10384,12302,12159,13735,12389,13381,15067)
```

这里不再检验指标的正态分布性，用户可以参考4.1.2自行检验。如果指标不服从正态分布，则可以对其做正态化处理，具体参考4.1.2。

（1）已知推广前pv指标的均值为10000时

如果已知推广前pv指标的均值为10000，通过如下代码可得知广告推广是否对网站pv有明显影响。

```
sol<-t.test(pv.new,mu=10000); sol
        One Sample t-test
data:  pv.new
```

```
t = 5.6513, df = 9, p-value = 0.0003131
alternative hypothesis: true mean is not equal to 10000
95 percent confidence interval:
 11514.58 13536.42
sample estimates:
mean of x
  12525.5
```

上面的代码主要用于检验原假设：推广后的 pv 指标均值等于推广前的 pv 指标均值 10000。使用 t.test 函数产生的结果 p-value（可以通过 sol$p.value 来直接调用）等于 0.0003131。它小于 0.05，说明原假设不成立，即推广前后的 pv 指标均值不相等，本次推广对网站流量 pv 有明显的影响。

（2）未知推广前 pv 指标的均值，并选取相同数目的两组数据对比时

下面选取推广前后各 15 天的 pv 数据，并检验原假设：推广后 pv 均值等于推广前 pv 均值。代码如下：

```
pv.old<-c(10231,12874,11229,9870,11804,11652,13259,11891,12876,14562,12933,1354
8,15230,13815,15766)
pv.new<-c(10740,13368,11730,10384,12302,12159,13735,12389,13381,15067,13415,140
34,15729,14327,16267)
t.test(pv.new,pv.old,paired=TRUE) $p.value
[1] 6.882981e-25
```

对比具有两组相同数目的样本数据时，可以把 paired 参数设置 TRUE，也可以直接检验 0 是否等于 pv.old-pv.new 指标均值。代码如下：

```
t.test(pv.old-pv.new)$p.value
[1] 6.882981e-25
```

由于 p-value 远远小于 0.05，因此推广后的 pv 均值并不等于推广前 pv 均值，即本次推广明显影响了网站的流量指标。

（3）未知推广前 pv 指标的均值，并选取不同数目的两组数据对比时

假设选取推广前 20 天的 pv 数据和推广后 15 天的 pv 数据进行对比。代码如下：

```
pv.old<-c(10231,12874,11229,9870,11804,11652,13259,11891,12876,14562,12933,1354
8,15230,13815,15766, 10500,12509,12737,11216,14117)
pv.new<-c(10740,13368,11730,10384,12302,12159,13735,12389,13381,15067,13415,140
34,15729,14327,16267)
t.test(pv.old,pv.new)$p.value
[1] 0.270269
```

此时不能再把 paired 参数设置为 TRUE。p-value 值 0.270269 大于 0.05，这与做成对检验的结果 6.882981e-25 差异很大，这表明两种检验方式存在巨大差异。一般而言，对比成对的两组指标时，使用 paired=TRUE 形式做 T 检验得到的 p-value 更小，判断更可靠。

注意

在条件允许的情况下，最好选取具有相同数目的两组数据做成对检验。

4.3.4 相同指标内的多组数据对比：单因素方差分析

4.3.3 节介绍了在不同情况下 2 组相同指标间如何进行对比。如果要扩展到多组指标间的对比，则需要使用单因素方差分析。例如，研究不同用户年龄段是否对客单价有影响时，实际是在检验一个多元分类变量是否对一个连续变量有明显影响。

1. 单因素方差分析原理

假设分类变量 A 有 r 个水平，依次是 $A1$，$A2$，$A3$，…，Ar。而被监控的指标 x 在变量 A 的不同水平下均服从正态分布（具有正态性），同时，每组指标 x 又具有相同的方差（方差齐次性），如表 4-7 所示。

表 4-7 单因素方差的样本示例

A 的水平	指标 x 的样本	服从的总体
$A1$	$x11$ $x12$ $x13$ $x14$ … $x1n1$	$N(\mu_1, \sigma^2)$
$A2$	$x21$ $x22$ $x23$ $x24$ … $x2n1$	$N(\mu_2, \sigma^2)$
	… … …	
Ar	$xr1$ $xr2$ $xr3$ $xr4$ … $xrn1$	$N(\mu_r, \sigma^2)$

则该指标差异产生的原因可以分为以下几种：

- 由指标 x 的随机性而产生的误差。此时使用组内方差 S_E 来衡量。
- 由分类变量 A 的不同取值而引起的差异。此时使用组间方差 S_A 来衡量。
- 看如下公式：

$$S_T=S_E+S_A=\sum_{i=1}^{r}\sum_{j=1}^{n_i}(x_{ij}-\overline{x})^2,\ \overline{x}=\frac{1}{n}\sum_{i=1}^{r}\sum_{j=1}^{n_i}x_{ij}$$

$$S_E=\sum_{i=1}^{r}\sum_{j=1}^{n_i}(x_{ij}-\overline{x}_i)^2,\ \overline{x}_i=\frac{1}{n_i}\sum_{j=1}^{n_i}x_{ij}$$

$$S_A=\sum_{i=1}^{r}\sum_{j=1}^{n_i}(\overline{x}_i-\overline{x})^2=\sum_{i=1}^{r}n_i(\overline{x}_i-\overline{x})^2$$

在上述公式中：

- 组内方差 S_E：由于方差分析存在正态性前提，即在分类变量 A 的水平值取 i 时，其指标服从正态分布 $N(\mu_i, \sigma^2)$，所以此时指标内的差异是由随机误差引起的。其随机误差的平方和是$\sum_{j=1}^{n_i}(x_{ij}-\overline{x}_i)^2$。而组内方差 S_E 表示不同水平下指标 x 的随机误差平方和之总和。
- 组间方差 S_A：$\sum_{i=1}^{r}n_i(\overline{x}_i-\overline{x})^2$表示在 Ai 水平下，指标样本的均值与所有样本的均值差异之和。而组间方差 S_A 则是所有水平下，样本均值差异之和的总和。

下面使用 4.3.2 节中的假设检验思想来进行假设。

- 原假设 H0：不同水平下，指标 x 无显著性差异。

❑ H1：不同水平下，指标 x 有显著性差异。

如果 H0 成立，则统计量$\frac{S_A/(r-1)}{S_E/(n-r)}$服从 $F(r-1,n-r)$ 分布（构建统计量的过程请参考表 4-8）。其中 $r-1$ 称为处理间自由度，$n-r$ 称为处理内自由度。

这里通过计算概率 p-value，并通过其与显著性 alpha 的大小关系来检验 H0 是否成立。

表 4-8 构建统计量

差异来源	自由度	差异平方和	均方	构建统计量
分类变量 A	处理间自由度 $r-1$	组间方差 S_A	$\frac{S_A}{r-1}$	$\frac{S_A/(r-1)}{S_E/(n-r)}$
随机性误差	处理内自由度 $n-r$	组内方差 S_E	$\frac{S_E}{n-r}$	
总和	$(r-1)+(n-r)=n-1$	总方差 S_T		

通过假设检验判断 H1，即可得到“不同水平下指标 x 有显著性差异”的结论，可是并不能得到“每组水平间的指标 x 均有显著性差异”。此时，要对数据样本做多重 T 检验，以便进一步确定哪些水平间的指标有显著差别。

2. 单因素方差分析的步骤

一般情况下使用 R 语言做单因素方差分析的步骤如图 4-24 所示。

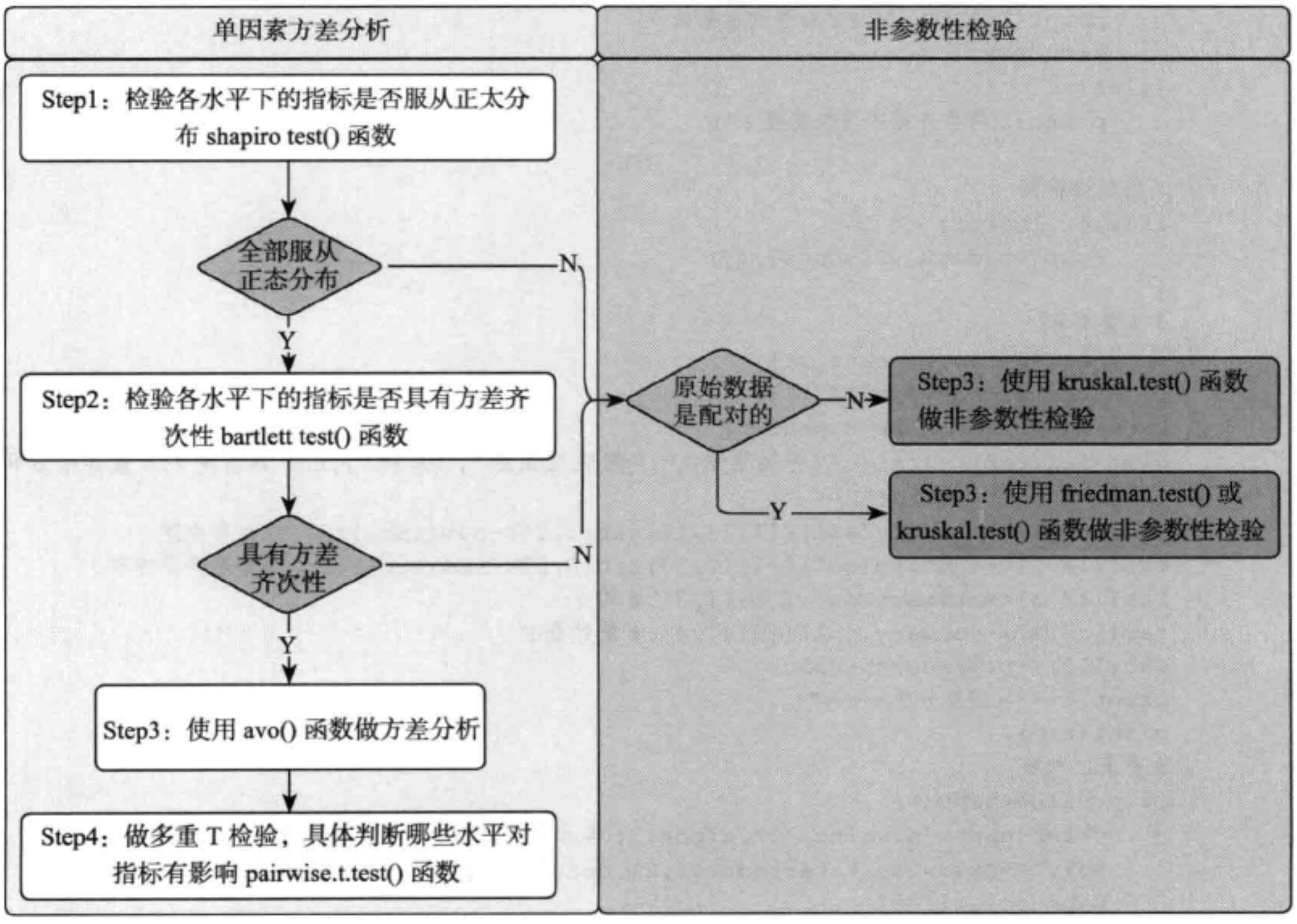


图 4-24 单因素方差分析的步骤

step1 和 step2 分别检验数据是否满足做正常方差分析的前提条件。如果不满足，则使用 friedman.test() 或 kruskal.test() 做非参数性检验。否则使用 aov() 做正常的方差分析，并对数据样本做多重 T 检验。方差分析代码如下：

```
aov.func<-function(data,alpha=0.05,plot.logic=T,p.adjust.mod="holm"){
    error<-FALSE
    data[,1]<-as.factor(data[,1])
    group.levles<- as.numeric(levels(data[,1])) #分类变量的水平取值
    #检验不同水平下指标的正态性
    for(i in group.levles){
        if(shapiro.test(data[which(data[,1]==i),2])$p.value<alpha){
            print(paste("ERROR:第",i,"组数据不服从正态分布"))
            error<-TRUE
        }
}
if(error){
    return()
}else{
    print("符合正态性前提!")
}
#检验不同水平下指标的方差齐次性
if(bartlett.test(data[,2]~data[,1])$p.value<alpha){
    print("ERROR:符合方差齐次性前提")
    return()
}else{
    print("符合方差齐次性前提!")
}
#绘制线箱图
if(plot.logic){
    boxplot(data[,2]~data[,1])
}
#方差分析
sol<-aov(data[,2]~data[,1])
    p.value<-summary(sol)[[1]][,5][1]
    tab<-matrix(NA,nrow=3,ncol=5)
    dimnames(tab)<-list(c("分类变量A","随机性误差","总和"),c("自由度","差异平方和
","均方","统计量F","p-value"))
    tab[1:2,1]<-summary(sol)[[1]][,1];tab[3,1]<-sum(tab[1:2,1])#自由度
    tab[1:2,2]<-summary(sol)[[1]][,2];tab[3,2]<-sum(tab[1:2,2])#差异平方和
    tab[1:2,3]<-summary(sol)[[1]][,3];#均方
    tab[1:2,4]<-summary(sol)[[1]][,4];#统计量F
    tab[1,5]<-p.value;#p-value
    print("====方差分析====")
    print(tab)
    #多重T检验
    if(p.value<alpha){
        print(paste(p.value,"<",alpha,":各水平下的指标有明显差别!"))
        sol.t<-pairwise.t.test(data[,2],data[,1],p.adjust.method=p.adjust.mod)
        tab<-sol.t[[3]]
        tab[which((sol.t[[3]]>alpha)==TRUE)]<-"无差别"
```

```
        tab[which((sol.t[[3]]>alpha)==FALSE)]<-" 显著差别 "
        print("==== 多重 T 检验 ====")
        print(tab)
    }else{
        print(paste(p.value,">",alpha,": 各水平下的指标无明显差别 !"))
    }

    return(sol)
}
```

自定义函数 aov.func 的参数说明如表 4-9 所示。

表 4-9 aov.func 函数的参数说明

参数	说明
data	可以是 matrix 或 data.frame 两种对象 第一列表示不同水平下的取值，第二列是对应的指标数据。例如：kpi<-c（rnorm（20,80,5）,rnorm（25,195,5）,rnorm（23,256,5）） group<-c（rep（1,20）,rep（2,25）,rep（3,23）） data<-data.frame（group,kpi） 1 81.68284 1 81.58639 1 81.37128 - - - 2 201.32094 2 192.99844 - - - 3 257.11924 3 254.21143
alpha	显著性水平，默认是 0.05
plot.logic	如果取 TRUE（默认时），则绘制 data 的线箱图 如果取 FALSE，则不绘制线箱图
p.adjust.mod	设置在做多重 T 检验时，修正 p-value 的使用方法

（1）正态性检验

在验证不同水平下的指标是否服从正态分布时，使用的核心函数是 shapiro.test()，这在 4.1.2 中已讲解，这里不再详述。只要有一组指标不服从正态分布，就打印“ERROR: 第 <i> 组数据不服从正态分布”。

（2）方差齐次性检验

在验证不同水平下指标数据的方差是否相同时，使用的核心函数是 bartlett.test()。如果得到的 p-value 大于显著水平 alpha，则表示符合方差齐次性。如果小于 alpha，则表示不符合方差齐次性。代码如下：

```
if(bartlett.test(data[,2]~data[,1])$p.value<alpha){
      print("ERROR: 符合方差齐次性前提 ")
      return()
}else{
      print(" 符合方差齐次性前提 !")
}
```

（3）方差分析 aov()

上面代码中的核心函数为：

```
sol<-aov(data[,2]~data[,1])
```

之后以表 4-8 的形式输出结果。例如：

```
[1] "====方差分析===="
            自由度 差异平方和         均方  统计量F       p-value
分类变量A        1  336515.95 336515.9533 402.0111 3.789226e-33
随机性误差      81   67803.59    837.0813       NA            NA
总和            82  404319.54          NA       NA            NA
[1] "3.78922610503873e-33 < 0.05 :各水平下的指标有明显差别!"
```

aov 函数的输入数据可以是向量形式，也可以是数据框形式。例如：

```
kpi<-c(rnorm(20,80,5),rnorm(25,195,5),rnorm(23,256,5))
group<-factor(rep(1:3,c(20,25,23)))
data<-data.frame(group,kpi)
aov(kpi~group,data=data)  # 以数据框形式

aov(data[,2]~data[,1])    # 以向量形式
```

注意，无论以哪种形式输入数据，符号～右边表示分类变量，符号～左边表示因变量，即欲研究的指标。

（4）多重 T 检验

如果经过 aov 函数检验后，可判定出“各水平下的指标有明显差别”，那么，还需要使用 pairwise.t.test() 函数依次对每两个不同水平下的指标做 T 检验。

```
sol.t<-pairwise.t.test(data[,2],data[,1],p.adjust.method=p.adjust.mod)
```

其中 p.adjust.method 参数用于设置修正 P-value 的方法。其取值可为 "holm"、"hochberg"、"hommel"、"bonferroni"、"BH"、"BY"、"fdr"、"none"。之所以要对多重 T 检验做修正，主要是因为在多次重复使用 T 检验时会极大增加产生第一类错误的概率，即“判定出有显著性差异”的结论不会很可靠。而经过修正后，产生的 P-value 值更大，结论就会更可靠。其输出结果如图 4-25 所示。

上面是对有 1 ～ 4 个不同水平的指标做多重 T 检验的结果。可见在水平取值为 3 和 4 时的指标无明显差别，其余的均有显著差别。

```
[1] "====多重T检验===="
  1             2             3
2 "显著差别" NA            NA
3 "显著差别" "显著差别" NA
4 "显著差别" "显著差别" "无差别"
```

图 4-25 输出结果

（5）函数输出

如果指标数据满足正态性和方差齐次性，则函数输出由 aov 函数返回的对象，即代码中的 sol。否则返回 NULL，之后可以使用非参数性检验，参见后面的 4.3.4 节。

3. 扩展：非参数性检验

如果样本数据不满足单因素方差分析的前提，就只能使用非参数性检验（事先不用检验指标的正态性和方差齐次性等前提条件）来判定不同水平间指标是否具有显著性差别。

（1）指标数据是配对的

例如，要对如表 4-10 所示的数据做非参数性检验。

表 4-10 配对型样本示例

A 的水平	指标 x 的样本
A1	x11, x12, x13, x14, x15, x16, x17, x18, x19, x110
A2	x21, x22, x23, x24, x25, x26, x27, x28, x29, x210
A3	x31, x32, x33, x34, x35, x36, x37, x38, x39, x310

如果在不同的水平下指标 x 的样本数均是 10，则可以使用如下代码：

```
a1<-c(x11, x12, x13, x14, x15, x16, x17, x18, x19, x110)
a2<-c(x21, x22, x23, x24, x25, x26, x27, x28, x29, x210)
a3<-c(x31, x32, x33, x34, x35, x36, x37, x38, x39, x310)
data<-matrix(c(a1,a2,a3),ncol=3)
friedman.test(data)
```

或使用：

```
data<-c(a1,a2,a3)
g<-gl(3,10);b<-gl(10,1,30)
friedman.test(data,g,b)
```

之后通过判断 $p.value 小于显著性 alpha，得到 a1、a2、a3 间有显著差异。如果 $p.value 大于显著性 alpha，则得到 a1、a2、a3 间无显著差异。

（2）指标数据不是配对的

例如，要对如表 4-11 所示的数据做非参数性检验。

表 4-11 非配对型样本示例

A 的水平	指标 x 的样本
A1	x11, x12, x13, x14, x15, x16, x17, x18
A2	x21, x22, x23, x24, x25, x26, x27, x28, x29, x210
A3	x31, x32, x33, x34, x35, x36, x37, x38, x39

在不同的水平下，如果指标 x 的样本数依次是 8、10、9，则可以使用如下语句：

```
a1<-c(x11, x12, x13, x14, x15, x16, x17, x18)
a2<-c(x21, x22, x23, x24, x25, x26, x27, x28, x29, x210)
a3<-c(x31, x32, x33, x34, x35, x36, x37, x38, x39)
kruskal.test(list(a1,a2,a3))
```

或使用：

```
data<-data.frame(kpi=c(a1,a2,a3),group=factor(rep(1:3,c(8,10,9))))
kruskal.test(kpi~group,data=data)
```

之后通过判断 $p.value 小于显著性 alpha，得到 a1、a2、a3 间有显著差异。如果 $p.value 大于显著性 alpha，则得到 a1、a2、a3 间无显著差异。

对于非参数检验需要使用 friedman.test() 或 kruskal.test() 来实现，函数的基本使用方式可以参考 aov 函数，这里不再详述。

4. 实例：不同年龄段对客单价的影响

以下数据是某电商网站 4 个年龄段的日客单价数据。下面以此为基础来讲解实例。

年龄	用户日客单价数据
<25 岁	79 79 87 79 71 84 82 82 85 82 88 81 71 76 76 81 75 72 83 76
26 ～ 30 岁	193 192 191 181 191 192 187 191 196 196 196 196 192 194 192 189 196 202 197 206 199 191 194 195 196
31 ～ 35 岁	252 252 255 264 254 252 253 260 258 258 259 254 251 250 257 250 257 261 246 252 258 259 251
>36 岁	258 248 263 256 260 261 248 255 254 250 252 258 255 245 253

首先准备数据，代码如下：

```
    price1<-c(79, 79, 87, 79, 71, 84, 82, 82, 85, 82, 88 ,81 ,71 ,76 ,76, 81, 75,
72, 83 ,76)
    price2<-c(193, 192, 191, 181, 191, 192, 187 ,191, 196 ,196 ,196 ,196 ,192 ,194
,192 ,189 ,196 ,202 ,197 ,206 ,199 ,191 ,194 ,195 ,196)
    price3<-c(252, 252, 255, 264, 254, 252, 253, 260 ,258, 258 ,259 ,254, 251 ,250
,257 ,250, 257 ,261 , 246 ,252, 258 ,259 ,251)
    price4<-c(258, 248 ,263 ,256 ,260 ,261 ,248, 255, 254, 250, 252, 258, 255, 245,
253)
    price<-c(price1,price2,price3,price4)
    date=rep(1:4,c(20,25,23,15))
    data<-data.frame(group=date,x=price)
```

然后使用 4.3.4 节中的自定义函数 aov.func 来做单因素方差分析。代码如下：

```
    sol<-aov.func(data)
```

打印输出结果，如图 4-26 所示。

```
[1] "符合正态性前提!"
[1] "符合方差齐次性前提!"
[1] "====方差分析===="
             自由度 差异平方和         均方  统计量F       p-value
分类变量A         1  336515.95 336515.9533 402.0111 3.789226e-33
随机性误差       81   67803.59    837.0813       NA            NA
总和             82  404319.54          NA       NA            NA
[1] "3.78922610503873e-33 < 0.05 :各水平下的指标有明显差别!"
[1] "====多重T检验===="
  1           2           3
2 "显著差别" NA          NA
3 "显著差别" "显著差别" NA
4 "显著差别" "显著差别" "无差别"
```

图 4-26 打印结果

绘制出的线箱图如图 4-27 所示。

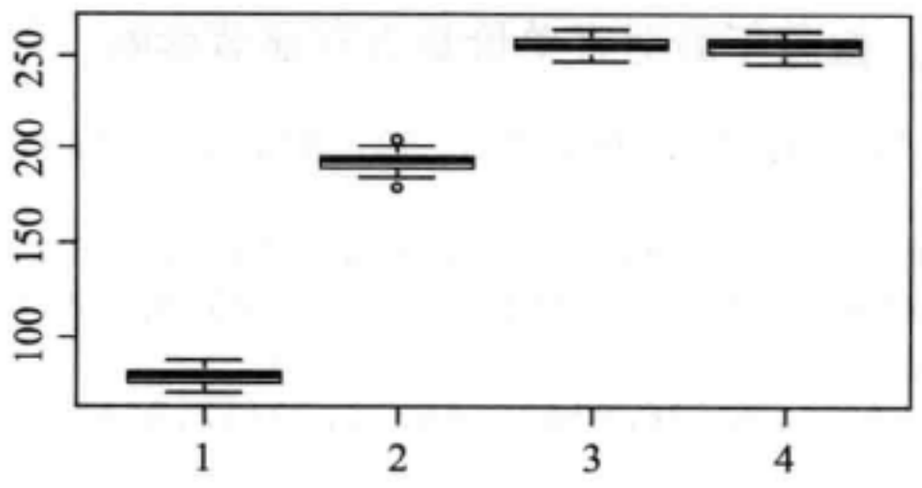


图 4-27 线箱图

可见不同年龄段对用户日客单价有显著的影响。其中小于 25 岁的用户客单价最低不到 100 元，26 ～ 30 岁的用户客单价居中，约为 200 元，31 ～ 35 岁和超过 36 岁的用户客单价无显著差异，约为 250 元左右。

实例中的数据符合正态性和方差齐次性，因此可以使用 aov 做方差分析。如果不符合方差分析条件，则直接使用如下语句：

```
kruskal.test(list(price1,price2,price3))
        Kruskal-Wallis rank sum test
data:   list(price1, price2, price3)
Kruskal-Wallis chi-squared = 59.4872, df = 2, p-value = 1.209e-13
```

仍能判断出不同年龄段的客单价有显著差别（p-value = 1.209e-13<0.05 ）。

4.3.5 单因素协方差分析

在介绍单因素方差分析时，只研究了分类变量 A 对指标 x 有影响的情况，然而在实际操作中，它很有可能受另一个连续变量影响。例如，在 4.3.4 节的实例中研究年龄因素对客单价的影响时，还要排除当天独立访问用户数对客单价引起的差异。

单因素协方差分析是指：在排除 UV 数据对客单价引起的差异后，再检验年龄对客单价是否有显著影响。我们把当日 UV 称为协变量。

表 4-12 在 4.3.4 节实例的基础上给出了当日 UV 的数据，涉及网站年龄段 – 日客单价 – 日独立访问用户数等数据。

表 4-12 数据集

年龄	定量指标：日客单价和当日 UV	
<25 岁	日客单价	79 79 87 79 71 84 82 82 85 82 88 81 71 76 76 81 75 72 83 76
	当日 UV	81011 78313 91522 80950 72560 87483 85403 83561 87223 85294 …
26 ～ 30 岁	日客单价	193 192 191 181 191 192 187 191 196 196 196 196 192 194 192 …
	当日 UV	82121 82608 82568 76671 79589 80357 77332 81449 83310 82537 …
31 ～ 35 岁	日客单价	252 252 255 264 254 252 253 260 258 258 259 254 251 250 257 …
	当日 UV	80155 81360 78110 85082 81288 81385 80840 84448 83051 84070 …
>36 岁	日客单价	258 248 263 256 260 261 248 255 254 250 252 258 255 245 253
	当日 UV	82067 80942 84601 82626 83405 84811 79412 83451 82907 80616 …

下面以 UV 为协变量，检验年龄对客单价是否有显著影响。代码如下：

```
    price1<-c(79, 79, 87, 79, 71, 84, 82, 82, 85, 82, 88 ,81 ,71 ,76 ,76, 81, 75,
72, 83 ,76)
    uv1<-c(81011, 78313, 91522, 80950, 72560, 87483, 85403, 83561, 87223, 85294,
88893, 82414, 76079, 80616, 78477, 85281, 77163, 74974, 85494, 79951)
                                                ...   ...   ...
    uv4<-c(82067,80942,84601,82626,83405,84811,79412,83451,82907,80616,80384,83260,
81707,79674,80317)
    price<-c(price1,price2,price3,price4)
    uv<-c(uv1,uv2,uv3,uv4)
    group=as.factor(rep(1:4,c(20,25,23,15)))
    data<-data.frame(age=group,price=price,uv=uv)
```

使用如下代码，绘制不同 group 水平下的散点图，以查看整体数据的表现情况。

```
coplot(price~uv|age,data=data,pch=16)
```

显示结果如图 4-28 所示。

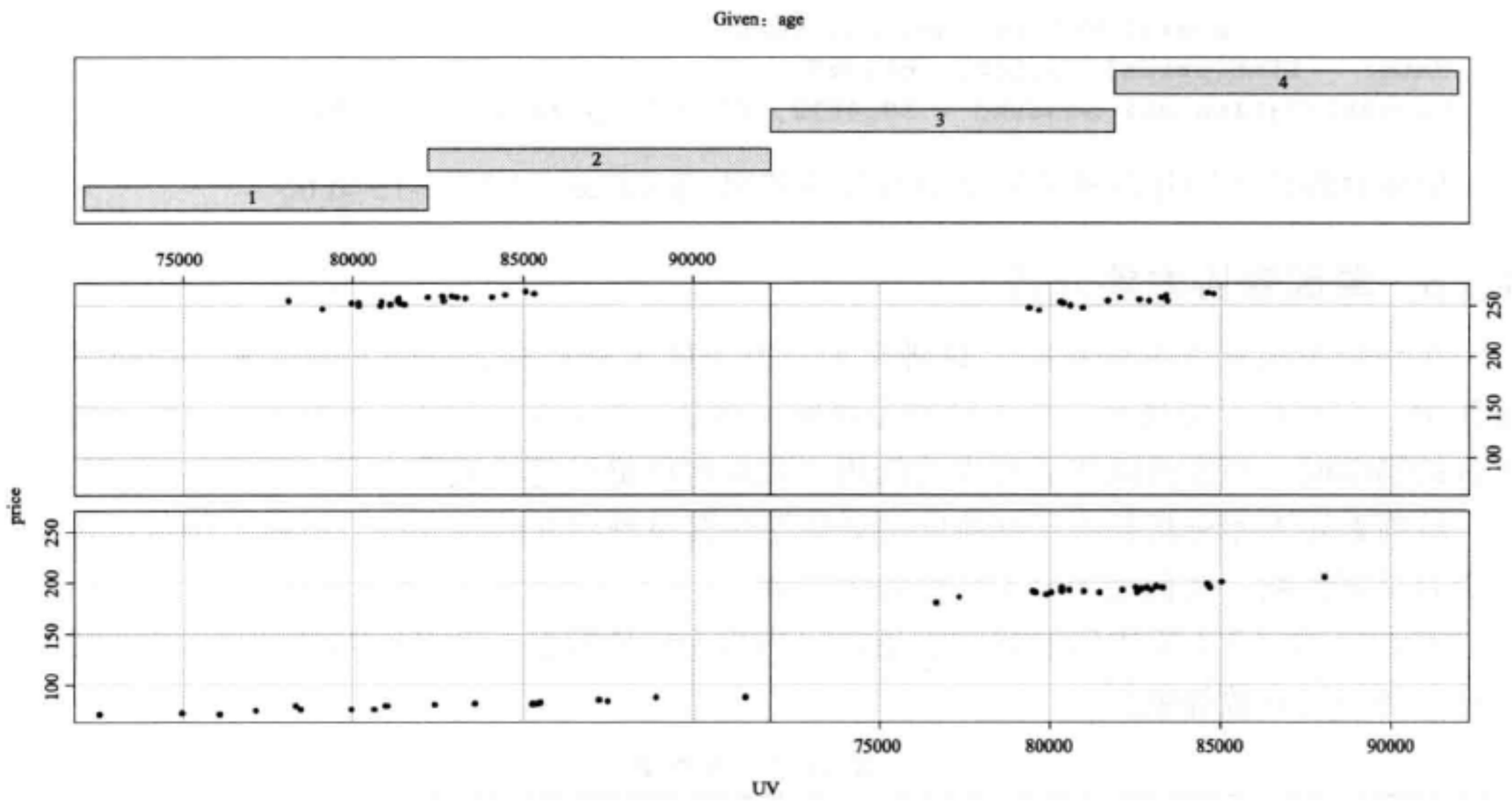


图 4-28　不同 group 水平下的散点图

1. 协方差前提

协方差分析的前提是不仅要有正态性和方差齐次性，还要满足无交互效应这一条件。

（1）正态性

正态性与 4.3.4 节的单因素方差分析并无区别，代码如下：

```
shapiro.test(subset(data,age==1)$price)
shapiro.test(subset(data,age==2)$price)
```

```
shapiro.test(subset(data,age==3)$price)
shapiro.test(subset(data,age==4)$price)
```

这里不再详述。

（2）方差齐次性

方差齐次性的检验也与4.3.4节的单因素方差分析相似，其核心函数为bartlett.test。代码如下：

```
bartlett.test(price~age,data=data)
            Bartlett test of homogeneity of variances
data:    price by age
Bartlett's K-squared = 0.6413, df = 3, p-value = 0.887
```

由结果可知，$p.value=0.887大于典型的显著性水平0.05，即符合方差齐次性条件。

（3）无交互效应

无交互效应在这里是指：age和uv指标对price指标并无交互效应。可以通过观察回归斜率是否显著相同（如果斜率相同则无交互效应）或者直接使用aov（连续变量price～协变量uv* 分类变量age）来判定。

下面先观察回归模型的斜率相同法。在图4-29中绘制4个年龄段下uv和price指标的散点图以及其线性回归线：price=a×uv+b。

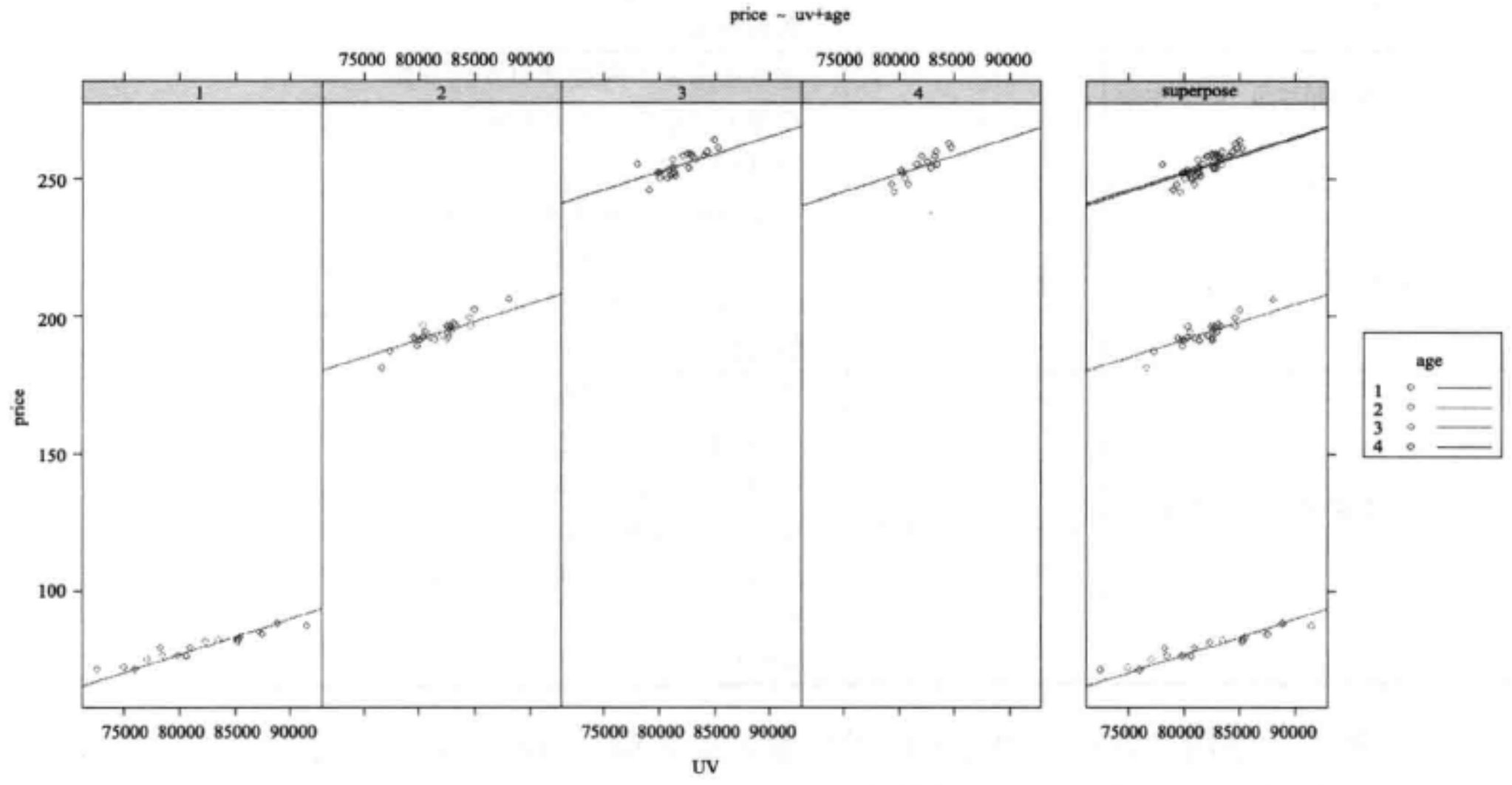


图4-29 4个年龄段下uv和price指标的散点图和线性回归线

观察得到，每组回归模型的直线是平行的，即满足无交互效应的前提。绘制图4-29的代码如下：

```
library(HH)
ancova(price~uv+age,data=data)
```

下面使用 aov 函数来判断“uv 和 age 的交互效应对客单价指标无显著性影响”。代码如下：

```
sol<-aov(price~uv*age,data=data)
summary(sol)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
uv	1	299	299	0.3528	0.5542
age	1	336909	336909	397.0069	<2e-16 ***
uv:age	1	70	70	0.0824	0.7748
Residuals	79	67041	849		

观察到 uv 和 group 交互效应（**uv:age**）的 p-value 为 0.7748，大于典型的显著性 0.05，它表示 uv 和 age 交互效应对客单价 price 指标无显著性影响，即满足无交互效应的前提。

2. 协方差分析

aov 函数的基本形式为 aov（formula,data=dataframe），其中通过对 formula 的设置可以分析单因素 / 多因素等方差。表 4-13 中的 y 表示预研究的因变量指标；x 表示可能影响 y 的协变量指标；A、B、C 表示分类变量指标。

表 4-13 formula 说明

分析方法	Formula 表达式	影响 y 差异的来源
单因素方差分析	$y \sim A$	分类变量 A 的不同水平引发的差异。 随机性误差
含单个协变量的单因素方差分析	$y \sim x+A$	连续变量 x 不同取值引发的差异。 控制 x 变量后，A 的不同水平引发的差异。 随机性误差
双因素方差分析	$y \sim A*B$	分类变量 A 不同水平引发的差异。 控制 A 后，B 的不同水平引发的差异。 控制 A 和 B 主效应时，A 与 B 的交互效应引发的差异。 随机性误差
含两个协变量的双因素方差分析	$y \sim x1+x2+A*B$	连续变量 $x1$ 不同取值引发的差异。 控制 $x1$ 变量后，$x2$ 的不同取值引发的差异。 控制 $x1$ 和 $x2$ 后，A 的不同水平引发的差异。 控制 $x1$、$x2$ 和 A 后，B 的不同水平引发的差异。 控制 $x1$、$x2$ 以及 A 和 B 主效应后，A 与 B 的交互效应引发的差异。 随机性误差

在本例中，使用“含单个协变量的单因素方差分析”方法，并把 x 设置为协变量 uv，A 设置为分类变量 age，y 是 price。代码如下：

```
sol<-aov(price~uv+age,data=data); summary(sol)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
uv	1	299	299	41.623	8.532e-09 ***
age	3	403459	134486	18696.127	< 2.2e-16 ***
Residuals	78	561	7		

上述代码第 2 行（age）的含义是控制 uv 变量后，age 的不同水平引发的差异，**Pr (>F)** 小于等于 2.2e-16 ***，表明在协变量 uv 下变量 age 对 price 有显著性影响。

4.4 分类型指标的对比

4.4.1 列联表分析

下面以某电影网站的 10 万条评分数据为例来介绍列联表函数的使用。具体的数据包下载地址为 http://vdisk.weibo.com/s/sFZzvownJIJ1，可使用如下代码加载该数据：

```
load("C:\\root\\data.read.RData")
```

其中数据框对象 data.frame.ratings 是用户投票数据，具体格式如图 4-30 所示。

	userid	moviesid	ratings	timestamp
1	1	1193	5	978300760
2	1	661	3	978302109
3	1	914	3	978301968
4	1	3408	4	978300275
5	1	2355	5	978824291
6	1	1197	3	978302268

图 4-30 加载后的用户投票数据

上面每条数据均记录了用户对某一个电影的投票数据：userid 是用户 ID，moviesid 是影片 ID，ratings 是用户的投票评分（1 ～ 5 分不等），timestamp 是时间戳，这里暂不使用。

数据框对象 data.frame.users 是用户信息数据，具体格式如图 4-31 所示。

	id	gender	age	occupation	zip.code
1	1	F	1	10	48067
2	2	M	56	16	70072
3	3	M	25	15	55117
4	4	M	45	7	02460
5	5	M	25	20	55455
6	6	F	50	9	55117

图 4-31 处理后的用户信息数据

上面每条数据表示一个用户的信息：id 是用户 ID，gender 是用户性别，age 是用户年龄，occupation 是用户职业，zip.code 代表用户城市地址，这里暂不使用。

通过 data.frame.ratings 中的 userid 和 data.frame.users 中的 id 将两者进行关联，合并两个数据框并最终组成数据框对象 data。代码如下：

```
data<-data.frame.ratings
data<-cbind(data,data.frame(gender=data.frame.users[1,2],
                            age=data.frame.users[1,3] ,
                            occupation=data.frame.users[1,4]))
userid=unique(data[,1])
```

```
    for(i in userid){
        data[which(data$userid==i),c("gender","age","occupation")]<-subset(data.
frame.users,id==i)[,c("gender","age","occupation")]
    }
    data$age<-round((data$age-5)/10)
```

上例中的核心语句为：

```
data[which(data$userid==i),c("gender","age","occupation")]<-subset(data.frame.
users,id==i)[,c("gender","age","occupation")]
```

它的意义是把 data 中 userid 等于 i 的各行中的 "gender"、"age"、"occupation" 列赋值为 data.frame.users 中 id 也等于 i 的 "gender"、"age"、"occupation" 数据，最后使用语句把年龄数据 age 划分年龄段：1 ～ 10 时转化后为 0，11 ～ 19 时转化后为 1，21 ～ 29 时转化后为 2，以此类推，51 ～ 59 时转化后为 5。

最终得到的 data 数据框如图 4-32 所示。

	userid	moviesid	ratings	timestamp	gender	age	occupation
1	1	1193	5	978300760	F	0	10
2	1	661	3	978302109	F	0	10
3	1	914	3	978301968	F	0	10
4	1	3408	4	978300275	F	0	10
5	1	2355	5	978824291	F	0	10
6	1	1197	3	978302268	F	0	10

图 4-32 最终的 data 数据框

1. 建立列联表

table 函数通过向量数据建立频数统计的列联表，xtabs 函数则可以直接通过数据框对象来建立列联表。

1）一维列联表：统计不同投票得分数的数目。代码如下：

```
table(data$ratings)
     1      2      3      4      5
 56174 107557 261197 348971 226310
```

2）二维列联表：统计不同性别下（行）的不同投票得分数（列）的数目。代码如下：

```
table(data$gender,data$ratings)
     1     2     3     4     5
F 13347 24548 62966 87033 58546
M 42827 83009 198231 261938 167764
xtabs(~gender +ratings,data=data)
       ratings
gender     1     2     3     4     5
     F 13347 24548 62966 87033 58546
     M 42827 83009 198231 261938 167764
```

3）三维列联表：统计用户性别 – 年龄段 – 投票得分的数目。代码如下：

```
table(data$gender,data$age,data$ratings)
, ,  = 1
        0     1      2      3      4     5
  F   731  3611   4881   2148   1656   320
  M  1507  9452  19017   6919   4701  1231
, ,  = 2
        0     1      2      3      4     5
  F   736  5561   9345   4773   3602   531
  M  2247 16512  35472  15480  10828  2470
              …省略…
xtabs(~gender+age +ratings,data=data)
, , ratings = 1
        age
gender     0     1      2      3      4     5
     F  731  3611   4881   2148   1656   320
     M 1507  9452  19017   6919   4701  1231
, , ratings = 2
        age
gender     0     1      2      3      4     5
     F  736  5561   9345   4773   3602   531
     M 2247 16512  35472  15480  10828  2470
              …省略…
```

下面的 ftable 把 table 产生的 3 种维列联表以更紧凑的形式展示。

```
tab.3<-table(data$gender,data$age,data$ratings)
ftable(tab.3)
         1     2      3      4     5
F 0    731   736   2023   3036  2301
  1   3611  5561  12345  14452  9458
  2   4881  9345  23577  32551 20986
  3   2148  4773  12767  17866 11919
  4   1656  3602  10404  15725 10787
  5    320   531   1850   3403  3095
M 0   1507  2247   4357   5772  4501
  1   9452 16512  35256  45789 31100
  2  19017 35472  80710 104273 64744
  3   6919 15480  40223  54117 32791
  4   4701 10828  30372  41093 26955
  5   1231  2470   7313  10894  7673
```

对于二维列联表，没有必要使用 ftable 函数处理。

2. 处理列联表

（1）prop.table 函数

prop.table 函数可以计算上面建立的列联表的比例值。代码如下：

```
tab<-table(data$gender,data$ratings)
prop.table(tab,1)
          1         2         3         4         5
```

```
F    0.05415923    0.09961045    0.25550235    0.35316101    0.23756695
M    0.05681714    0.11012525    0.26298641    0.34750434    0.22256686
```

prop.table（tab,1）语句在第 1 个变量 gender 的不同水平下，计算其余数据的比例值（其和为 1）。

```
prop.table(tab,2)
          1          2          3          4          5
F 0.2376010  0.2282325  0.2410671  0.2493989  0.2586982
M 0.7623990  0.7717675  0.7589329  0.7506011  0.7413018
```

prop.table（tab,2）语句在第 2 个变量 age 的不同水平下，计算其余数据的比例值（其和为 1）。

```
prop.table(tab)
          1          2          3          4          5
F 0.01334421 0.02454287 0.06295284 0.08701481 0.05853377
M 0.04281805 0.08299165 0.19818958 0.26188327 0.16772894
```

prop.table（tab）语句用于计算所有数据的比例值（所有数据总和为 1）。

（2）margin.table 函数

margin.table 函数可以计算上面建立的列联表的边际频数（边际求和）。代码如下：

```
margin.table(tab,1)
     F      M
246440 753769
```

margin.table（tab,1）语句计算第 1 个变量 gender 的频数（总和）。

```
margin.table(tab,2)
    1     2     3     4     5
56174 107557 261197 348971 226310
```

margin.table（tab,2）语句计算第 2 个变量 age 的频数（总和）。

```
margin.table(tab)
[1] 1000209
```

margin.table（tab）语句计算所有变量的总和。

（3）addmargins 函数

addmargins 函数用于添加上面建立的列联表的边际和。代码如下：

```
addmargins (tab,1)
          1          2          3          4          5
F     13347      24548      62966      87033      58546
M     42827      83009     198231     261938     167764
Sum   56174     107557     261197     348971     226310
```

addmargins（tab,1）语句计算第 1 个变量 gender 在不同水平下的总和。

```
addmargins (tab,2)
          1          2          3          4          5        Sum
  F   13347      24548      62966      87033      58546     246440
  M   42827      83009    1 98231     261938     167764     753769
```

addmargins（tab,2）语句计算第 2 个变量 ratings 在不同水平下的总和。

```
addmargins (tab)
          1          2          3          4          5        Sum
  F   13347      24548      62966      87033      58546     246440
  M   42827      83009     198231     261938     167764     753769
  Sum 56174     107557     261197     348971     226310    1000209
```

addmargins（tab）语句计算每个变量不同水平下的总和。

3. 列联表的可视化

使用 vcd 包中的 mosaic 函数可以对联列表进行可视化展示，其数据输入的格式与 xtabs 函数较为相似。下面是涉及用户性别 – 投票得分 – 年龄的可视化代码。

```
library(vcd)
mosaic(~gender+ratings+age,data=data,shade=T,legend=T)
```

显示结果如图 4-33 所示。

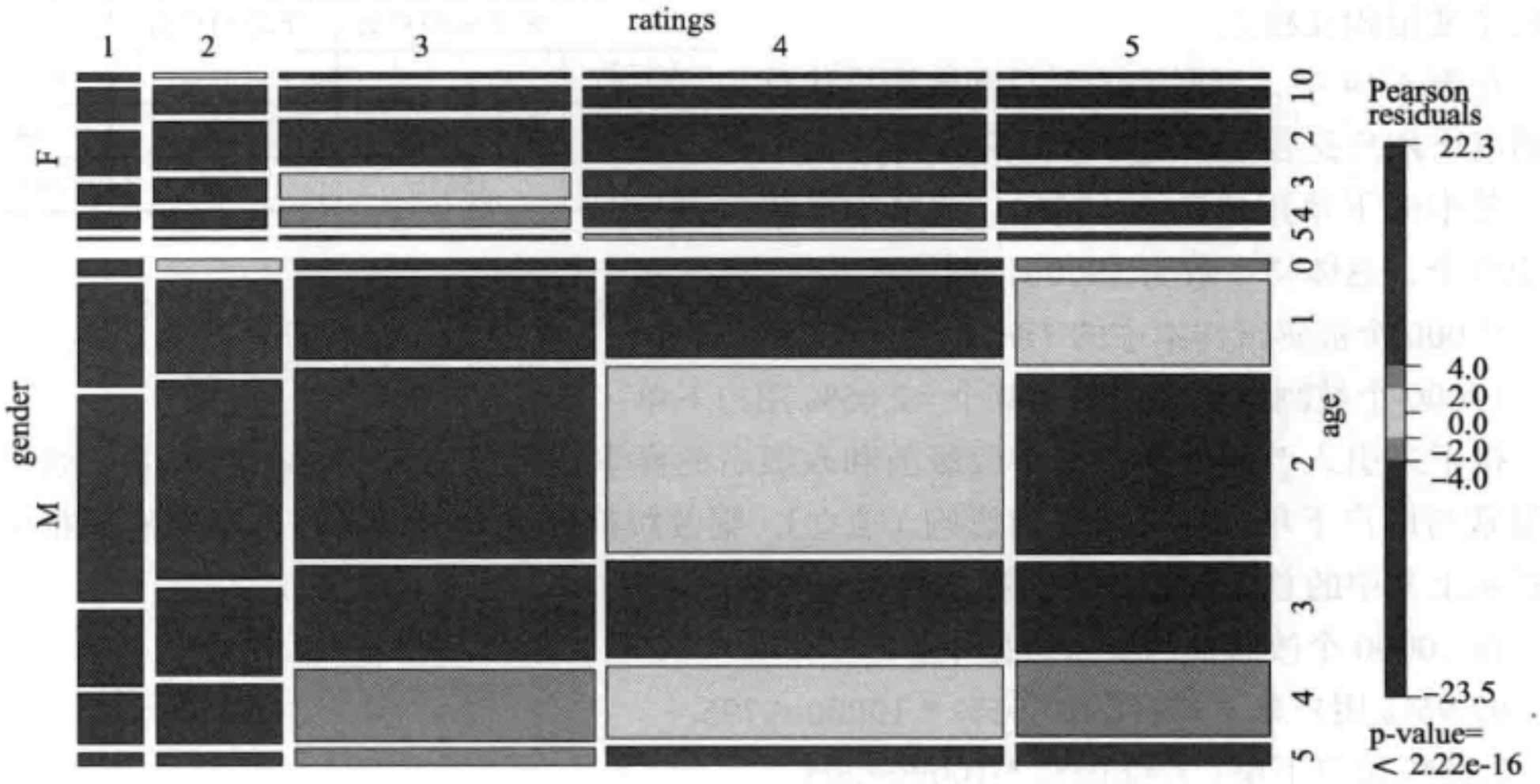


图 4-33 结果展示

图 4-33 中的颜色表示列联表中各部分的实际频数 f_o 和期望频数 f_e 的接近程度（根据拟合模型的皮尔逊残差值对图形进行配色），其中灰色表示比较接近，蓝色和红色表示有显著差异。

也可以使用联列表对象作为其数据的输入：mosaic（< 联列表对象 >,shade=T,legend=T）。

下面的代码同上面的代码等价。

```
library(vcd)
data.tab<- xtabs(~gender +ratings+age,data=data)
mosaic(data.tab,shade=T,legend=T)
```

4.4.2 卡方独立性检验

检验不同分类变量间是否有显著性关系，最常用的方法是卡方检验。它的典型应用之一是：对比分类变量在不同水平下的转化率型数据是否有显著变化。例如，网站改版前后的订单转化率分别为2.0%和2.11%，看上去貌似本次改版使转化率指标提升了0.11个百分点。然而考虑到改版前后独立访问用户量分别为10000和9000，那是否还能得出“改版后转化率有所提升”的结论呢？下面介绍如何解决以上问题。

1. 卡方检验原理

这里以改版前后是否对订单转化率产生影响为例来说明卡方检验的原理。由于订单转化率=完成下单的用户数/（没有下单+完成下单的用户数），所以可以把研究订单转化率问题转化为研究用户下单和不下单的问题。即研究改版这个因素（水平取值为：改版前和改版后）对用户下单是否有影响。如果并无影响，则称这两个变量相互独立。

表4-14 列联表原始表格

	未下单用户数	下单用户数	总和
改版前			10000
改版后			9000
总和	18610	390	19000

在表4-14中，列联表的列总和数据描述总体情况下用户是否下单的分布情况。

其中未下单用户数为18610，下单用户数为390个，总体样本数为19000，则：

19000个被测试样本中的18610个=97.95%用户未下单

19000个被测试样本中的390个=2.05%用户下单

接下来引入改版因素，其中改版前和改版后的样本数目分别为10000和9000。如果改版因素与用户下单之间并无任何影响（独立），则改版前和改版后用户是否下单的分布也应该服从上文中的总体分布，我们称之为期望频数f_e。

在10000个改版前的样本数据中：

97.95%用户未下单：f_e=97.95% * 10000=9795

2.05%用户下单：f_e=2.05% * 10000=205

9000个改版前的样本数据中：

97.95%用户未下单：f_e=97.95% * 9000=8815.5

2.05%用户下单：f_e=2.05% * 9000=184.5

具体的期望频数如表4-15所示。

然而实际频数f_o却如表4-16所示。

表 4-15 改版因素与用户下单之间并无任何影响（独立）情况下的期望频数 f_e

	未下单用户数	下单用户数	总和
改版前	10000*18610/19000=9795	10000*390/19000=205	10000
改版后	9000*18610/19000=8815.5	9000*390/19000=184.5	9000
总和	18610	390	19000

表 4-16 实际频数 f_o

	未下单用户数	下单用户数	总和
改版前	9800	200	10000
改版后	8810	190	9000
总和	18610	390	19000

卡方统计就是测量实际频数 f_o 和期望频数 f_e 的拟合程度。其公式如下：

$$卡方 = x^2 = \sum \frac{(f_o - f_e)^2}{f_e}$$

如果两个变量之间并无影响，则 f_o 和 f_e 越接近，统计量（卡方）越小。考虑用一个临界值 3.84 来作为其最大的阈值。选取该值的原因是：自由度 df 为 1 且显著水平 alpha 为 0.05 时，卡方分布的值为 3.84。而自由度为 1 的原因是：上例是 2 行 2 列的列联表，因此可以确定自由度 df=（R−1）(C−1）=（2−1）(2−1）=1。

在本例中，卡方等于（9800−9795）^{2}/9795+（200−205）^{2}/205+（8810−8815.5）^{2}/8815.5+（190−184.5）^{2}/184.5=0.2909

由于该值小于临界值 3.84，所以可以得出结论：网站改版和用户是否下单之间并无影响，即本次改版实际并没有影响到用户的下单行为。而这与单纯地通过对比改版前后转化率而得到两个变量之间有关联的结论恰恰相反。

在 R 语言中，常计算该卡方统计量下的 p-value 概率值并同显著水平 alpha 进行对比来判断结论。

如果 p-vlaue 大于 alpha（典型取值是 0.05），则两个变量之间没有影响（相互独立）。

如果 p-vlaue 小于 alpha（典型取值是 0.05），则两个变量之间有关系（相互不独立）。

2. chisq.test 函数

chisq.test 函数可以对二维列联表进行卡方独立检验，以判断行变量和列变量是否相互独立。其基本形式为：chisq.test（x,correct=TRUE,…），参数说明如表 4-17 所示。

表 4-17 chisq.test 函数的参数说明

参数	说明	
x	取矩阵或列联表对象，表示二维列联表数据	
	x 是矩阵对象，上例中可使用： rbind（c（9800,200）,c（8810,190）） [,1] [,2] [1,] 9800 200 [2,] 8810 190	x 是列联表对象。 由 table 函数和 xtabs 函数建立
correct	如果为 TRUE（默认情况下），则在计算卡方统计量时应有连续性校正。 如果为 FALSE，则正常计算卡方统计量	

在返回的对象中，Sp.value 属性为 p-value 值，如果其大于显著性水平 0.05，则可得出行列变量互相独立的结论，如果其小于显著性水平 0.05，则可得出行列变量互相影响的结论。

以下是使用 chisq.test 函数进行卡方检验的代码，引用了 4.4.2 节中改版订单转化率的数据，并得出与实例相同的结论（X-squared = 0.2909）。

```
chisq.test(rbind(c(9800,200),c(8810,190)),correct=FALSE)
```

```
        Pearson's Chi-squared test
data:  rbind(c(9800, 200), c(8810, 190))
X-squared = 0.2909, df = 1, p-value = 0.5897
```

下面的代码使用 4.4.1 节中的 mosaic 函数，对其进行可视化。

```
library(vcd)
data.tab<- as.table(rbind(c(9800,200),c(8810,190)))
dimnames(data.tab)<-list(c("改版前","改版后"),c("未下单用户数","下单用户数"))
mosaic(data.tab,shade=T,legend=T)
```

绘制的图形如图 4-34 所示。

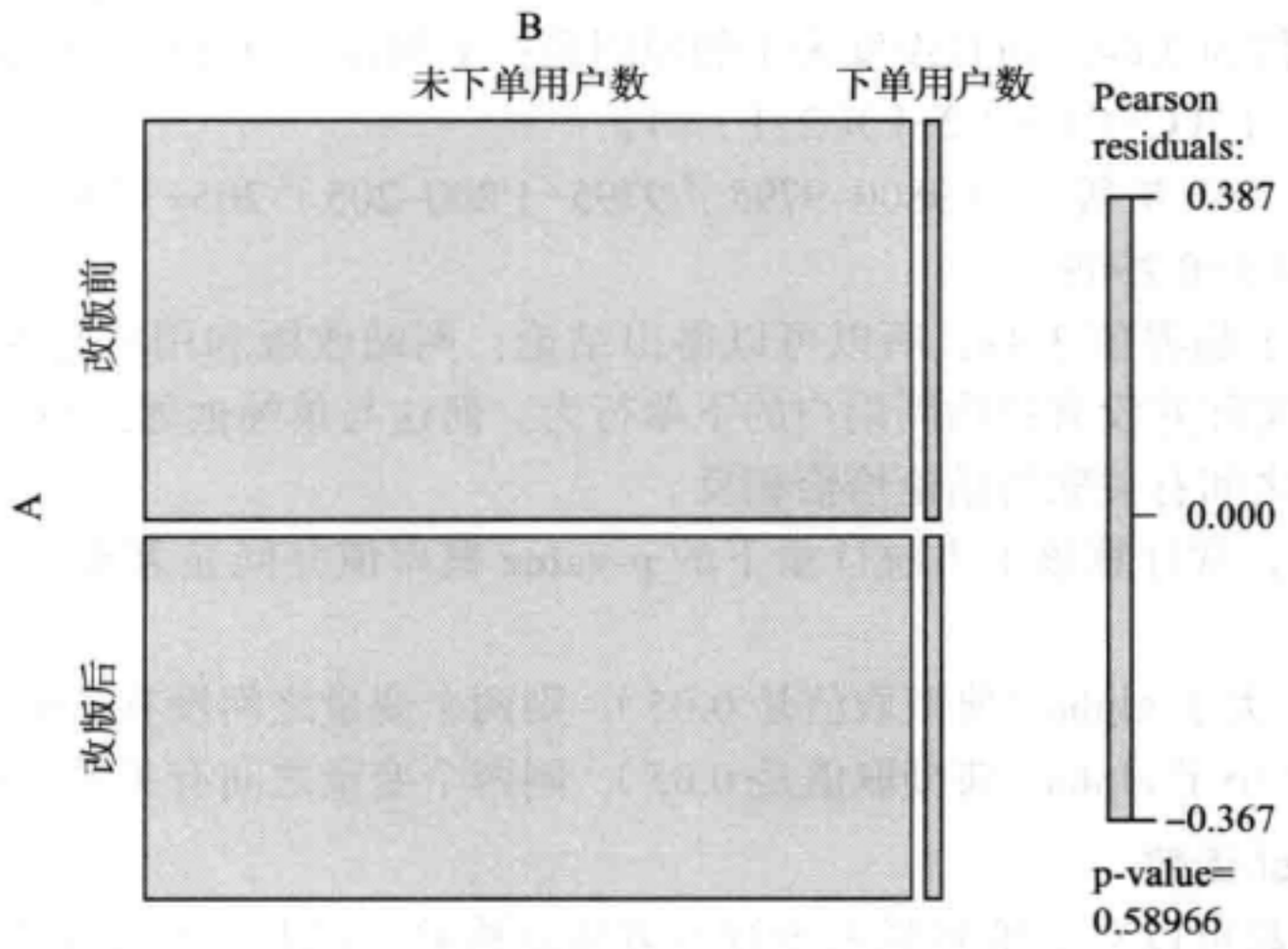


图 4-34 可视化结果展示

其中，灰色表示各个部分的实际频数 f_o 和期望频数 f_e 十分接近（并无显著差异），这也间接说明改版因素与转化率是独立的。

3. 实例：不同网龄的用户对网站的满意度

上面使用网站改版前后订单转化率的数据来讲解卡方检验的原理。这里以不同网龄用户对网站满意度调查的问卷数据为例，进一步讲解卡方分析的应用。具体数据如下：

数据	网龄	很不满意	基本满意	较满意	很满意
	<1 年	70	24	80	82

数据	网龄	很不满意	基本满意	较满意	很满意
	1年～2年	42	48	104	125
	2年～5年	33	28	181	113
	>5年	37	18	154	92

首先研究用户的网龄是否对网站满意度有所影响。代码如下：

```
data1<-c(70,24,80,82)# 网龄小于1年的用户，对网站满意度的评分数据
data2<-c(42,48,104,125)# 网龄在1～2年的用户，对网站满意度的评分数据
data3<-c(33,28,181,113)# 网龄在2～5年的用户，对网站满意度的评分数据
data4<-c(37,18,154,92)# 网龄大于5年的用户，对网站满意度的评分数据
data<-rbind(data1,data2,data3,data4)
sol<-chisq.test(data,correct=FALSE)
sol$p.value
[1] 5.730802e-14
```

计算得到 p.value 为 5.730802e-14，小于典型显著水平 0.05，说明用户的网龄与网站满意度是互相影响的（非独立性），并使用 mosaic 函数进行可视化。代码如下：

```
data.tab<- as.table(data)
dimnames(data.tab) <-list(c("<1年","1-2年","2-5年",">5年"),c("很不满意","基本满意","较满意","很满意"))
mosaic(data.tab,shade=T,legend=T)
```

显示结果如图 4-35 所示。

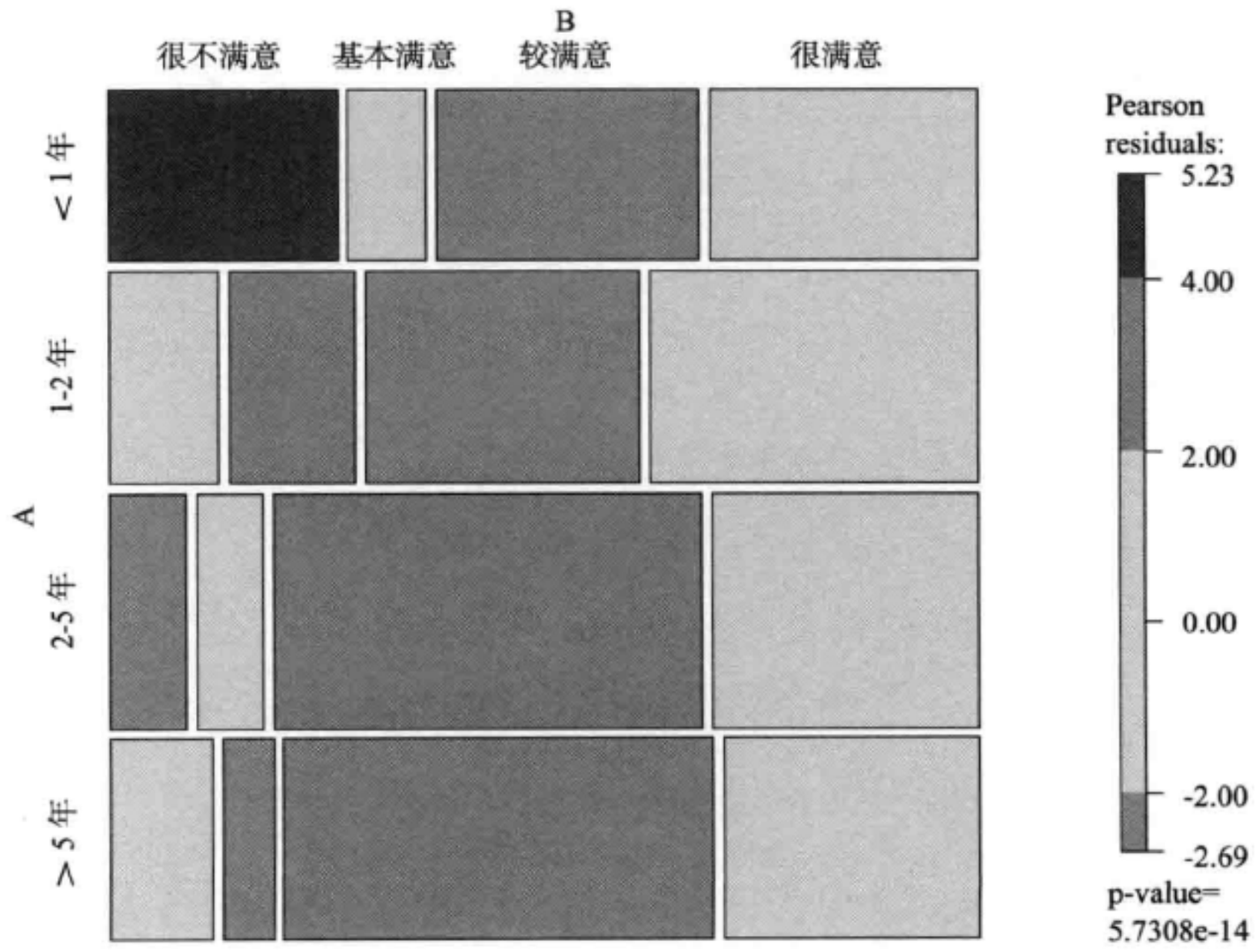


图 4-35 结果显示

在图 4-35 中，左上角为深蓝色（其拟合模型的皮尔逊残差值为 4.00 ～ 5.23），表明网龄小于 1 年且很不满意的实际频数 f_o 和期望频数 f_e 有较大差异。由于其皮尔逊残差值为正，所以它也表明：网龄小于 1 年的用户比其他网龄段的用户更加不满意网站。

图 4-35 中第三列各部分的含义如表 4-18 所示。

表 4-18 结论说明

网龄	皮尔逊残差值	结论
<1 年	-2.69 ～ -2.00（粉色）	网龄小于 1 年和 1 ～ 2 年的用户有较少的人选择“较满意”，而网龄在 2 ～ 5 年和在 5 年以上的用户有较多的人选择“较满意”，即网龄在 2 年以上的用户更喜欢本网站
1 ～ 2 年	-2.00 ～ -2.69（粉色）	
2 ～ 5 年	2.00 ～ 4.00（浅蓝色）	
>5 年	2.00 ～ 4.00（浅蓝色）	

第 5 章

时间序列分析

网站的日常指标多是以日为周期计算的，网站分析人员在观察日常指标时，不仅要监控其波动是否异常，还要发现其中的趋势，以便对指标进行预测。时间序列分析就是建立指标与时间轴关系的模型的方法。

5.1 时间序列

时间序列对象是一种专为时间序列分析而设计的对象类型，其中不仅包含指标的数值，还包括时间轴信息。时间序列对象和一般的向量相似，在很多情况下，时间序列分析函数处理的数据既可以使用向量，也可以使用时间序列对象。

在 R 语言中，ts（< 向量对象 >）可以把一个向量转化为一个时间序列对象。下面以某电商网站 2006 年 1 月至 2012 年 12 月间每月的销量数据为例进行说明，该数据存储于向量 sales.data 中，其长度是 84。下面把它转化为时间向量。代码如下：

```
sales.ts<-ts(sales.data,frequency=12,start=c(2006,1))
```

其中 start 参数表示时间起始点，设置为（2006,1）表示 2006 年 1 月。frequency 参数是频率，这里设置为 12，即把一年分为 12 份，每个样本表示 1 个月。最后的数据格式如图 5-1 所示。

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2006	3752.96	3714.74	4349.61	3566.34	5021.82	6423.48	7600.60	19756.21	2499.81	5198.24	7225.14	4806.03
2007	5900.88	4951.34	6179.12	4752.15	5496.43	5835.10	12600.08	28541.72	4717.02	5702.63	9957.58	5304.78
2008	6492.43	6630.80	7349.62	8176.62	8573.17	9690.50	15151.84	34061.01	5921.10	5814.58	12421.25	6369.77
2009	7609.12	7224.75	8121.22	7979.25	8093.06	8476.70	17914.66	30114.41	4826.64	6470.23	9638.77	8821.17
2010	8722.37	10209.48	11276.55	12552.22	11637.39	13606.89	21822.11	45060.69	7615.03	9849.69	14558.40	11587.33
2011	9332.56	13082.09	16732.78	19888.61	23933.38	25391.35	36024.80	80721.71	10243.24	11266.88	21826.84	17357.33
2012	15997.79	18601.53	26155.15	28586.52	30505.41	30821.33	46634.38	104660.67	19612.99	19056.05	21596.62	20611.17

图 5-1 数据格式

下面的语句绘制出销量指标时间序列的散点图，如图 5-2 所示。

```
plot(sales.ts,main="月销售数据：2006年1月--2012年12月",type="b",
     pch=16,ylab="月销量（万）",xlab="月份")
```

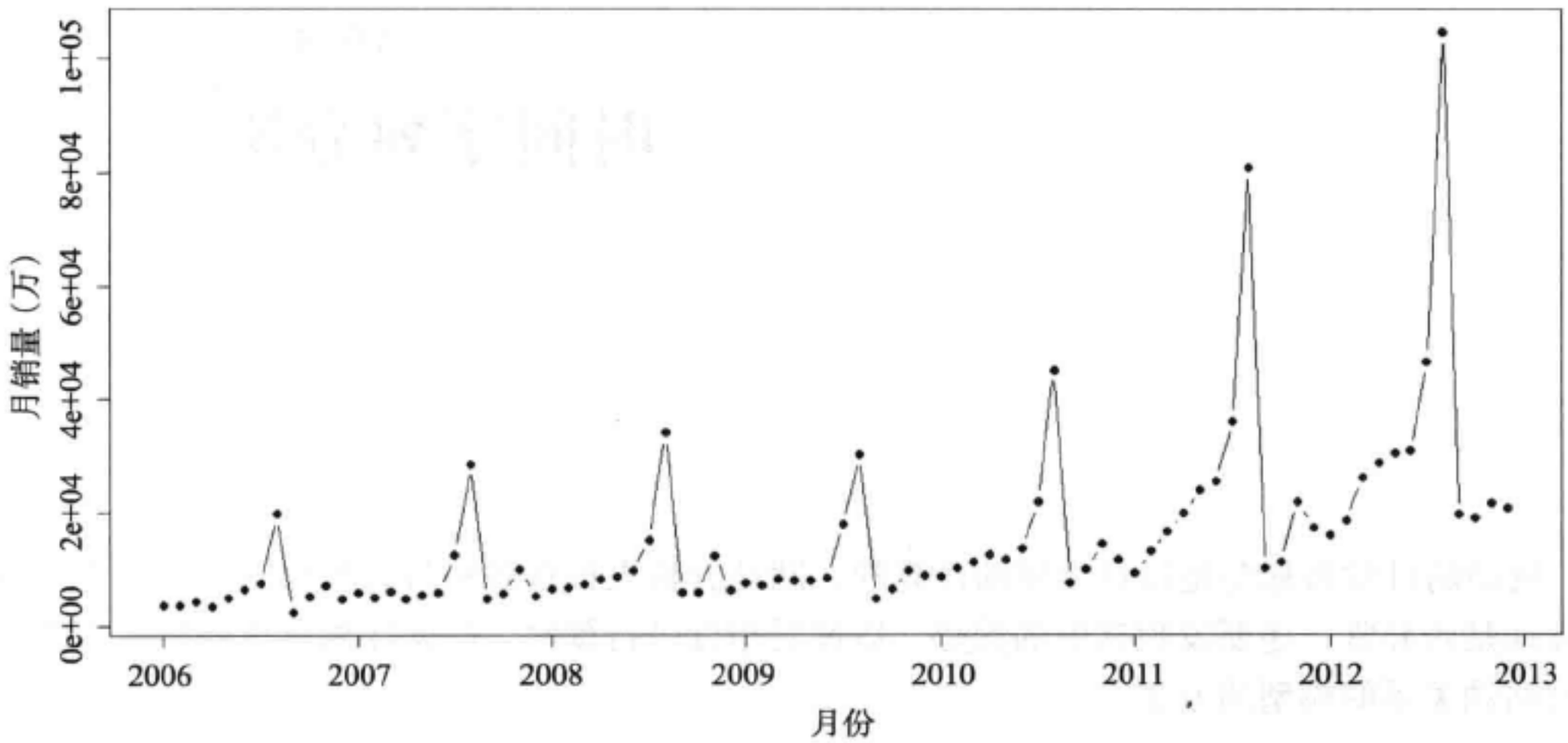


图 5-2 月销售数据

其中 x 轴的尺度标识根据 ts() 函数确定。从图 5-2 中至少可以看出如下特征：

- 随着时间的增长，销量数据总体呈增长趋势。
- 每年的 7 月和 8 月是销售的高峰期，即它具有一定的周期性。

实际上，很多复杂的时间序列都可以使用长期趋势 + 周期性变化 + 随机变化的形式来表示（也可使用累乘形式：长期趋势 × 周期性变化 × 随机变化）。而时间序列分析的主要目的就是找到这些趋势和周期性变化，并建立一个模型，使模型内的计算值与真实样本值间的残差最小。

decompose 函数可以把时间序列分解成长期趋势和周期性变化，其基本参数如表 5-1 所示。

表 5-1 decompose 函数的参数

参数	说明
x	时间序列数据，注意 x 必须是设定了频率（实际是周期长度）的序列。decompose 函数依据 x 的频率来分解长期趋势和周期趋势
type	分解类型，如果取值为 "additive"，则分解为累加形式：长期趋势 + 周期性变化 + 随机变化。如果取值为 "multiplicative"，则分解为累乘形式：长期趋势 × 周期性变化 × 随机变化。默认使用 "additive" 累加形式

使用如下代码对上面的销售数据进行季节性变化分析。

```
sol<-decompose(sales.ts)
plot(sol)
```

上述代码绘制的季节性变化分析图如图 5-3 所示。

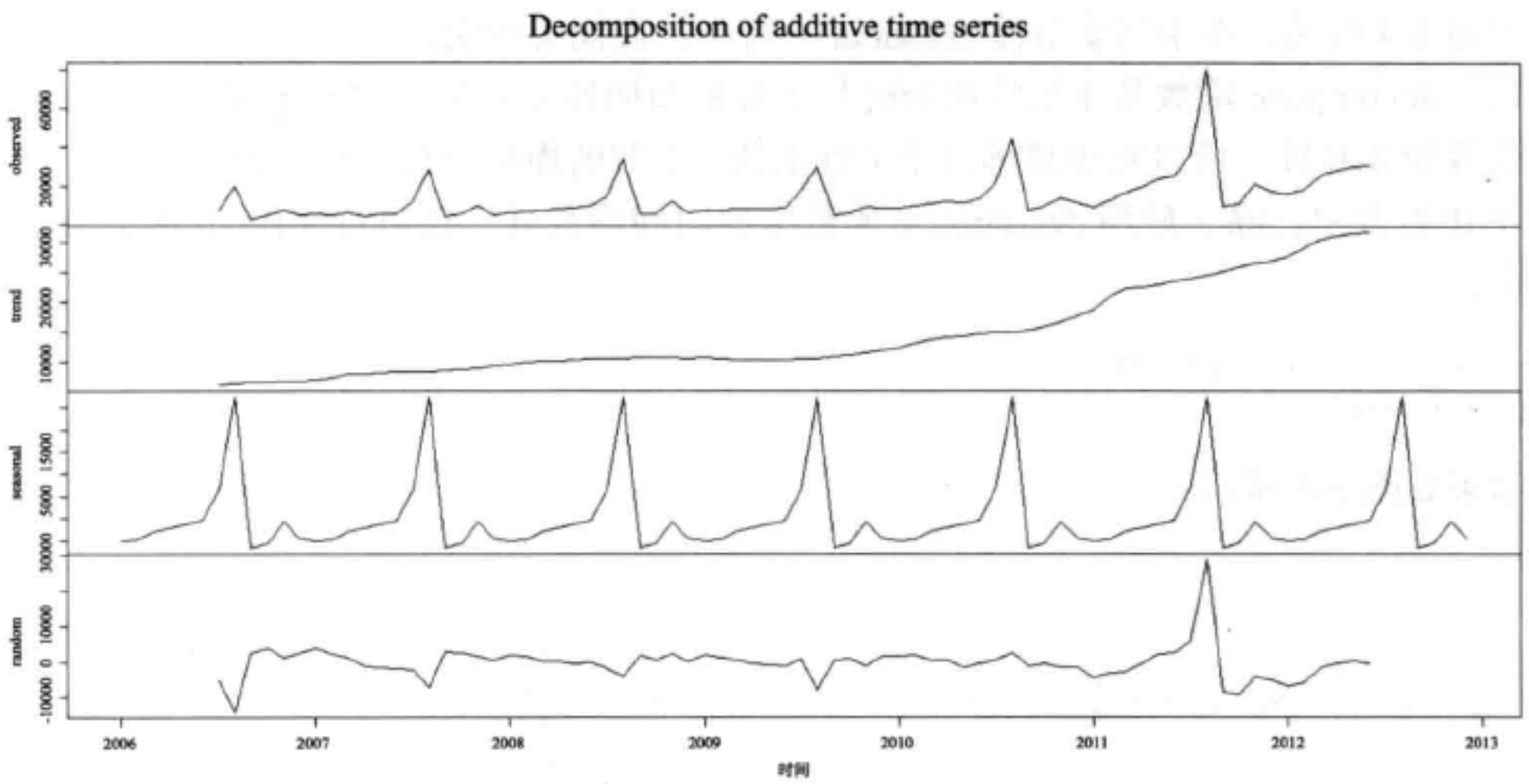


图 5-3 季节性变化分析结果展示图

在图 5-3 中，trend 是时间序列趋势（可以直接通过 sol$trend 获取），seasonal 是季节性周期变化（可以直接通过 sol$seasonal 获取），random 是随机误差（可以直接通过 sol$random 获取）。注意，季节变化的分析结果与时间序列 sales.ts 的频率有关系，本节最开始提到的 sales.ts，其频率被设置为 12，因此这里分析的季节性就是长为 12 的周期变化。如果把 sales.ts 的频率设置为 3，并再次做季节性分析：

```
sales.ts<-ts(sales.data,frequency=3,start=c(2006,1))
sol<-decompose(sales.ts)
plot(sol)
```

则上面代码输出的图形如图 5-4 所示。

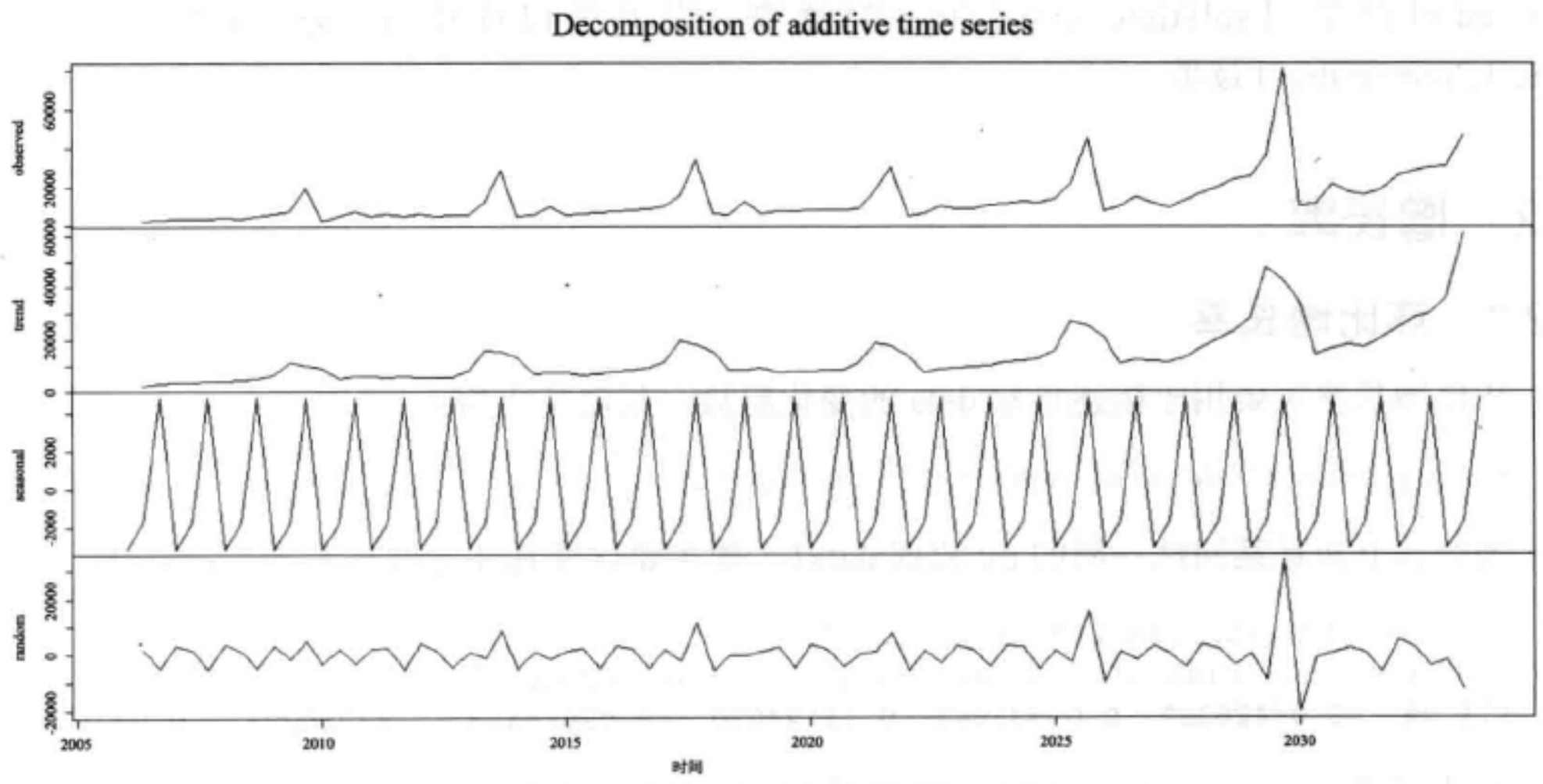


图 5-4 frequency=3 时的结果展示图

由图 5-4 可见，在中间季节性 seasonal 一行中，周期长度明显变小（实际是 3）。由此可以看出，decompose 函数是在给定周期时长下分析周期性变化的，它不能探索性地分析出周期时长等相关参数。所以它也被称为季节性变化（季节的周期时长是固定的 12 个月）。

在 R 语言中，除了使用 decompose 函数分解时间序列外，还可以使用 stl 函数分解。语句如下：

```
sol<- stl(sales.ts,"per")
plot(sol)
```

结果如图 5-5 所示。

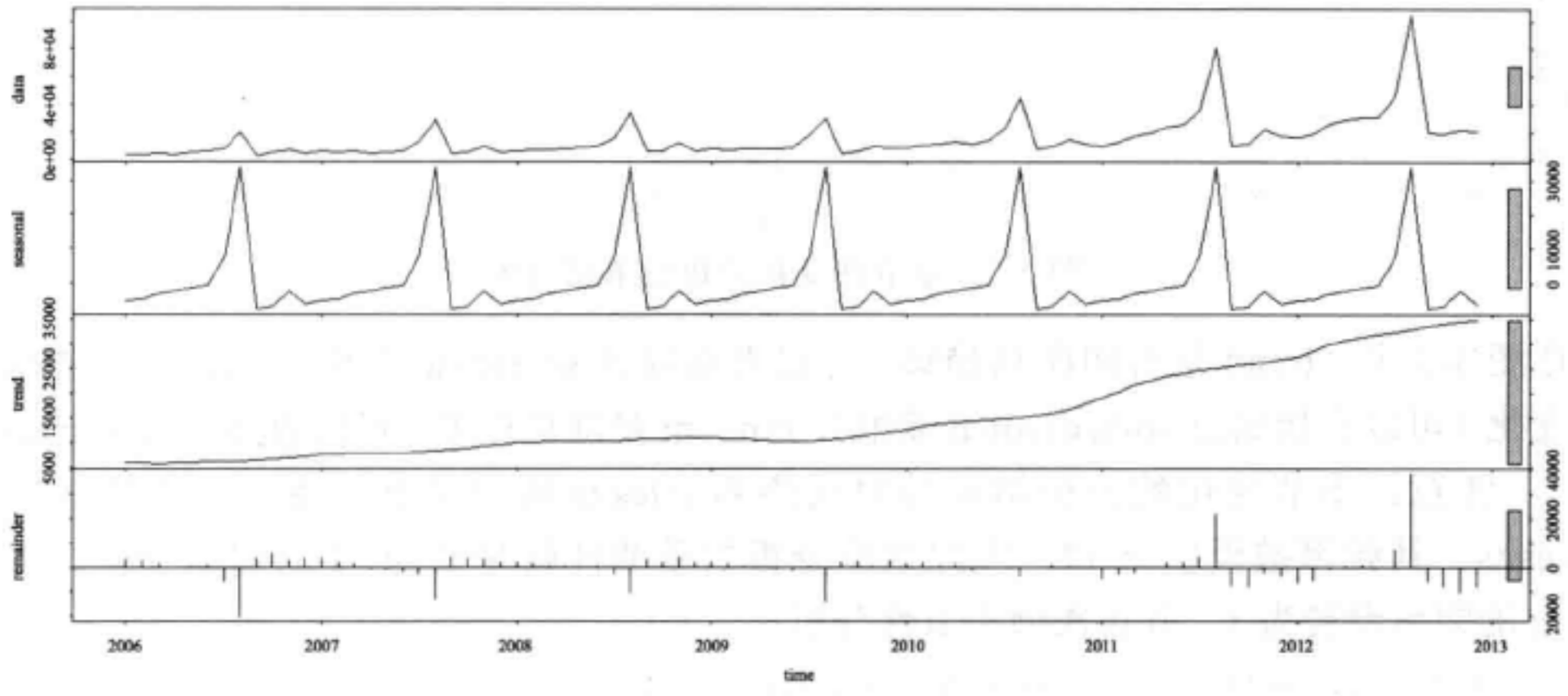


图 5-5 stl 函数结果展示图

其中周期性序列 seasonal 可以直接使用 sol$time.series[,"seasonal"] 读取。长期趋势 trend 可以使用 sol$time.series[,"trend"] 读取。误差可以使用 stl（sales.ts,"per"）$time.series[,"remainder"] 读取。

5.2 增长率

5.2.1 环比增长率

环比增长率主要用于描述指标 data 的变化趋势，其定义式为：

```
ring.growth[t]=data[t]-data[t-1]）/data[t-1]
```

例如，下面是某网站一周的 pv 数据 data1，现在要计算其环比增长率 ring.growth：

```
data1<-c(129,122,134,149,146,215,208)
ring.growth<- c(NA,diff(data1)/data1[1:(length(data1)-1)]); ring.growth
[1] NA  -0.05426357  0.09836066  0.11194030  -0.02013423  0.47260274  -0.03255814
```

因为计算 ring.growth[t=1] 时，需要使用 data[0]，而 data[0] 无数据，所以默认 ring.

growth[t=1] 等于 NA。

5.2.2 同比增长率

如果想避免周期性变化给数据分析带来的影响，可以使用同比增长率来进行分析。例如，一般会以周为单位来分析网站 pv 的周期性变化，由于周末的 pv 较大，因此不能直接比较星期一和星期六的 pv，而是使用 pv 的同比增长率（同比周期为 1 周）来描述原有的 pv 指标变化情况，这样就可以对比同一周内每天的 pv 指标大小。其定义格式如下：

```
same.per.growth[t]=(data[t]-data[t-T]) /data[t-T]
```

其中 T 是同比周期，即 data[t] 和 data[t-T] 分别是前后两个周期中对应的指标数据。

例如，下面分别是某网站上周和本周的 pv 数据 data0 和 data1，现要计算本周的同比增长率 same.per.growth。

```
data0<- c(113,134,123,145,137,196,187)
data1<-c(129,122,134,149,146,215,208)
same.per.growth<-(data1-data0)/data0; same.per.growth
[1]  0.14159292  -0.08955224  0.08943089  0.02758621  0.06569343  0.09693878  0.11229947
```

虽然本周六的 pv（215）大于本周一的 pv（129），但由同比增长率可知，本周六 pv 指标的表现情况（同比增长率为 0.097）并不及本周一（同比增长率为 0.142），从而避免周期性给指标间比较带来的困难。

5.3 移动平均

5.3.1 数学原理概述

移动平均是从“信号与分析系统学”中发展出来的一种典型滤波方法。其思想是使用近期的样本数据来预测当前和未来的数据。它可以把抖动的曲线平滑化，从而弱化随机变化和短期周期性变化的影响，进而更明显地显示出长期的变化趋势曲线。移动平均又分为简单移动平均和加权移动平均，下面分别介绍。

简单移动平均的公式如下：

$$y_{移动平均}[t]=(x[t-1]+x[t-2]+x[t-3]+x[t-4]+\cdots+x[t-n])/n$$

其中 $y_{移动平均}[t]$ 是变换后的现阶段预测值，$x[t-1]$ 是前期实际值，$x[t-n]$ 是前 n 期实际值，n 是移动平均的时期数，也可简称为移动平均的阶数。

加权移动平均是对原变量之前阶段的样本数据做线性变换，其公式如下：

$$y_{加权移动平均}[t]=a1*x[t-1]+a_2*x[t-2]+a_3*x[t-3]+a_4*x[t-4]+\cdots+a_n*x[t-n]$$

其中 $a_1+a_2+a_3+a_4+\cdots+a_n=1$

例如，有一个长度为 15 的序列 11 20 23 17 19 21 20 27 24 24 32 42 39 53 42，其标准差为 11.58077。现在使用 $y_{加权移动平均}[t]=0.5*x[t-1]+0.3*x[t-2]+0.2*x[t-3]$ 进行转化，得到数据

19.7 19.4 19.2 19.6 20.1 23.7 24.1 24.6 28.0 35.4 38.5 46.6，其标准差变为了 8.970267，很明显小于原数据的标准差。

图 5-6 中的黑色曲线是原始数据线，蓝色为转化后的数据线，可见蓝色线的抖动更小，同时并未改变原有的增长趋势，达到了消除随机变化并突出长期变化趋势的滤波目的。

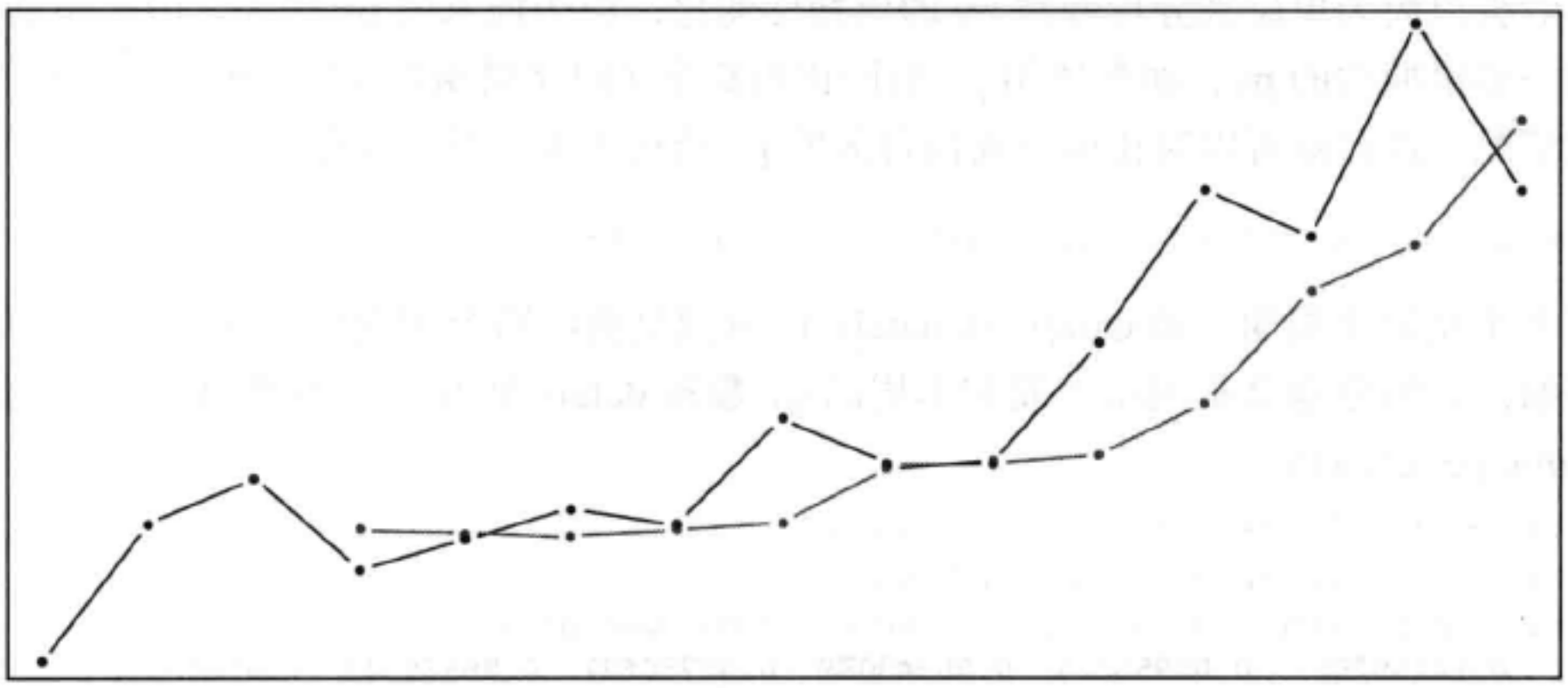


图 5-6 原始数据和加权移动平均处理后的数据对比图

5.3.2 filter 函数

filter 是线性过滤函数，其形式为：filter (x, filter, method = c ("convolution", "recursive") , sides = 2, ...)，相关参数说明如表 5-2 所示。

表 5-2 filter 函数参数说明

参数	说明	
x	待转化的数据	
method	过滤方法：convolution（卷积方法）和 recursive（递归方法）	
	卷积方法：使用 x 内部样本组成线性模型（系数由 filter 设置的），具体表达式参见 sides 说明。	递归方法：使用 y 内部样本以及当前阶段的 x 样本组成线性模型（系数由 filter 设置），$y_{递归}[t]=x[t]+\sum(a_i*y[t-i])$
filter	线性模型中的系数 a_i	
sides	method 取 recursive（递归方法）时，sides 没有意义。 method 取 convolution（卷积方法）时，sides 可以取以下值： 为 2（双边卷积）时，$y_{卷积}[t]=a_1*x[t+m]+\cdots\cdots+a_{(m+1)}*x[t]+\cdots\cdots+a_{(m+k+1)}*x[t-k]$ 为 1（单边卷积）时，$y_{卷积}[t]=a_1*x[t]+\cdots\cdots+a_{(k+1)}*x[t-k]$	

为了详细说明 filter 函数的使用方法，下面给出部分实例。

实例 1：双边卷积方法，代码如下：

```
x<-c(1:10);
y<-filter(x,filter=c(1,2,3,4), method ="convolution",sides=2);
Time Series:
Start = 1
```

```
End = 10
Frequency = 1
 [1] NA 20 30 40 50 60 70 80 NA NA
y<-c(y);y
[1] NA 20 30 40 50 60 70 80 NA NA
```

在上述代码中，method 参数设为 "convolution"，sides 参数设为 2，即使用双边卷积线性过滤方式。线性参数 filter 设置为 1、2、3、4，即线性模型是 $y[t]=1*x[t+2]+2*x[t+1]+3*x[t]+4*x[t-1]$。所以：

t=1 时，$y[1]= 1*x[3]+2*x[2]+3*x[1]+4*x[0]=1*3+2*2+3*1+4*NA=NA$。

t=2 时，$y[2]= 1*x[4]+2*x[3]+3*x[2]+4*x[1]=1*4+2*3+3*2+4*1=20$。

t=3 时，$y[3]= 1*x[5]+2*x[4]+3*x[3]+4*x[2]=1*5+2*4+3*3+4*2=30$。

……

t=9 时，$y[9]= 1*x[11]+2*x[10]+3*x[9]+4*x[8]=1*NA+2*10+3*9+4*8=NA$。

t=10 时，$y[10]= 1*x[12]+2*x[11]+3*x[10]+4*x[9]=1*NA+2*NA+3*10+4*9=NA$。

注意：$x[0]=x[11]=x[12]=NA$

filter 函数返回的对象 y 是一个时间序列，通过 $y=c(y)$ 把时间序列转化为向量形式。

实例 2：单边卷积方法，代码如下：

```
y<-filter(x,filter=c(1,2,3,4), method ="convolution",sides=1);y
Time Series:
Start = 1
End = 10
Frequency = 1
[1] NA NA NA 20 30 40 50 60 70 80
```

在上述代码中，method 参数设为 "convolution"，sides 参数设为 1，即使用单边卷积线性过滤方式。线性参数 filter 设置为 1、2、3、4，即线性模型是 $y[t]=1*x[t]+2*x[t-1]+3*x[t-2]+4*x[t-3]$。所以：

t=1 时，$y[1]= 1*x[1]+2*x[0]+3*x[-1]+4*x[-2]=1*1+2*NA+3*NA+4*NA=NA$。

t=2 时，$y[2]= 1*x[2]+2*x[1]+3*x[0]+4*x[-1]=1*2+2*1+3*NA+4*NA=NA$。

t=3 时，$y[3]= 1*x[3]+2*x[2]+3*x[1]+4*x[0]=1*3+2*2+3*1+4*NA=NA$。

t=4 时，$y[4]= 1*x[4]+2*x[3]+3*x[2]+4*x[1]=1*4+2*3+3*2+4*1=20$。

……

t=10 时，$y[10]= 1*x[10]+2*x[9]+3*x[8]+4*x[7]=1*10+2*9+3*8+4*7=80$。

注意：$x[0]=x[-1]=x[-2]=NA$

实例 3：递归方法，代码如下：

```
y<-filter(x,filter=c(1,2,3,4), method ="recursive");y
Time Series:
Start = 1
End = 10
```

```
Frequency = 1
 [1]    1    3    8   21   55  139  351  886 2234 5625
```

在上述代码中，method 参数设为 " recursive "，即使用递归线性过滤方式，此时的 sides 参数没有意义。线性参数 filter 设置为 1、2、3、4，即线性模型是 $y[t]= x[t]+1*y[t-1]+2*y[t-2]+3*y[t-3]+4*y[t-4]$。所以：

t=1 时，$y[1]= x[1]+1*y[0]+2*y[-1]+3*y[-2]+4*y[-3]=1+2*0+3*0+4*0=1$。

t=2 时，$y[2]= x[2]+1*y[1]+2*y[0]+3*y[-1]+4*y[-2]=2+1*1+2*0+3*0+4*0=3$。

t=3 时，$y[3]= x[3]+1*y[2]+2*y[1]+3*y[0]+4*y[-1]=3+1*3+2*1+3*0+4*0=8$。

t=4 时，$y[4]= x[4]+1*y[3]+2*y[2]+3*y[1]+4*y[0]=4+1*8+2*3+3*1+4*0=21$。

t=5 时，$y[5]= x[5]+1*y[4]+2*y[3]+3*y[2]+4*y[1]=5+1*21+2*8+3*3+4*1=55$。

…… ……

t=10 时，$y[10]= x[10]+1*y[9]+2*y[8]+3*y[7]+4*y[6]=10+1*2234+2*886+3*351+4*139=5625$。

注意：此时 $y[0]=y[-1]=y[-2]=y[-3]=0$，而非 NA。

5.3.3 R 语言实现

上面介绍了 filter 函数的使用方法，在 R 语言中，可以使用单边卷积线性过滤来分析趋势变化。还是以 5.1 节中的销售数据为例，下面对其做简单移动平均。代码如下：

```
plot(sales.ts,main="月销售数据：2006年1月--2012年12月",type="b",pch=16,ylab="月销量（万）",xlab="月份")
sales.filter1<- filter(sales.ts,c(0, rep(1/3,3)),"convolution",sides=1)
lines(sales.filter1, col="red",lwd=2, pch=16)
sales.filter2<- filter(sales.ts,c(0,rep(1/6,6)),"convolution",sides=1)
lines(sales.filter2,col="green",lwd=2, pch=16)
sales.filter3<- filter(sales.ts,c(0, rep(1/10,10)),"convolution",sides=1)
lines (sales.filter3,col="blue",lwd=2, pch=16)
```

结果展示如图 5-7 所示。

其中，红色曲线为 $y=（x[t-1]+x[t-2]+x[t-3]）/3$ 模型转化后的数据曲线，阶数 n=3。

绿色曲线为 $y=（x[t-1]+x[t-2]+\cdots+x[t-6]）/6$ 模型转化后的数据曲线，阶数 n=6。

蓝色曲线为 $y=（x[t-1]+x[t-2]+\cdots+x[t-10]）/10$ 模型转化后的数据曲线，阶数 n=10。

由图 5-7 可知，移动平均模型的阶数越大，曲线抖动越小。此时长期趋势更加明显，由蓝色曲线可见销售是趋于逐年上升的。有时认为越是近期的数据越具有价值，这时可以使用加权移动平均方式，例如，让 $x[t-1]$ 的加权系数大于 $x[t-2]$，然后以此类推。代码如下

```
plot(sales.ts,main="月销售数据：2006年1月--2012年12月",type="b",pch=16,ylab="月销量（万）",xlab="月份")
sales.filter1<- filter(sales.ts,c(0, c(1:3)/6),"convolution",sides=1)
lines(sales.filter1, col="red",lwd=2, pch=16)
sales.filter2<- filter(sales.ts,c(0, c(1:6)/21),"convolution",sides=1)
```

```
lines(sales.filter2,col="green",lwd=2, pch=16)
sales.filter3<- filter(sales.ts,c(0, c(1:10)/55),"convolution",sides=1)
lines (sales.filter3,col="blue",lwd=2, pch=16)
```

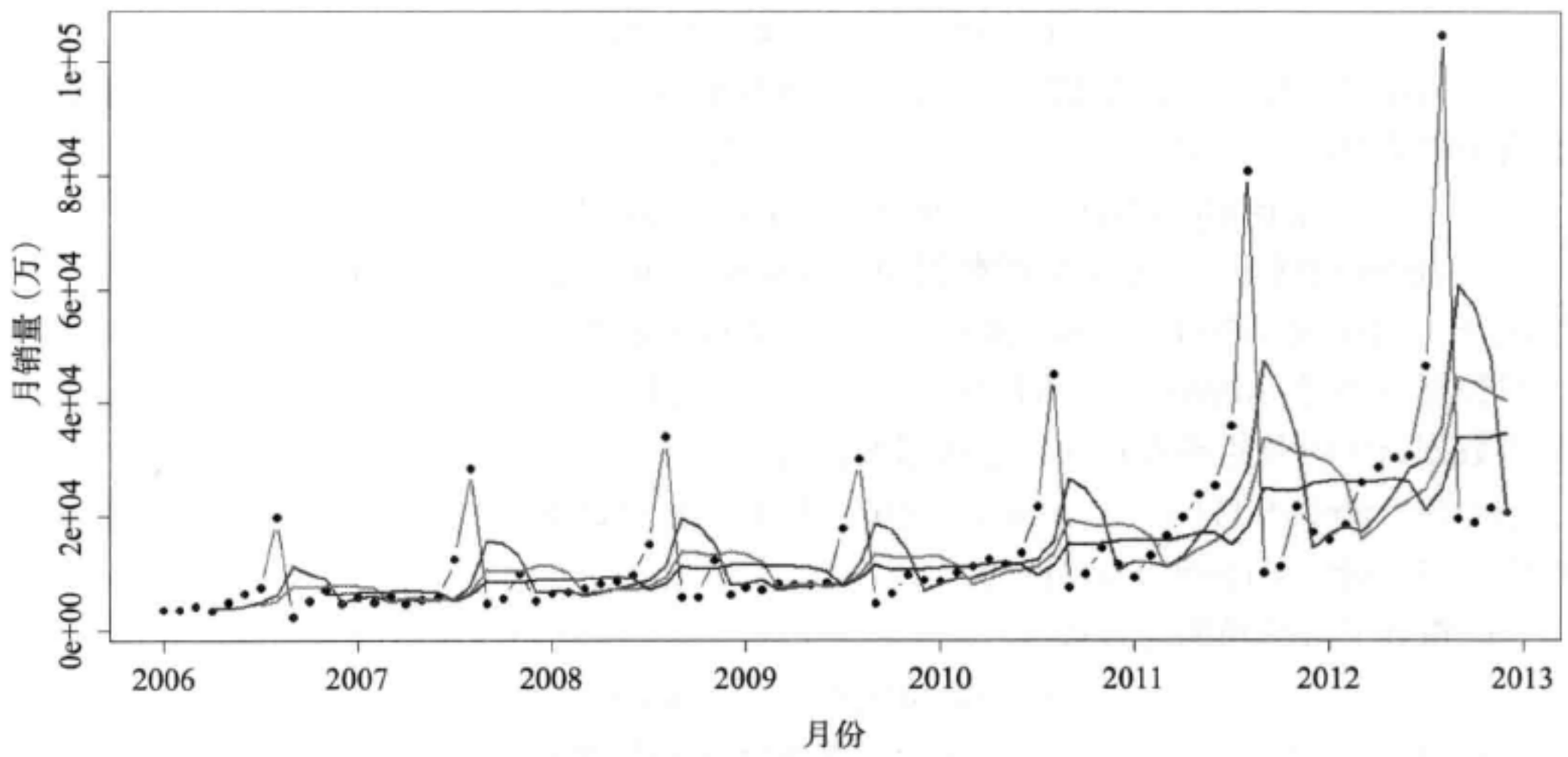


图 5-7　月销量数据和移动平均处理效果图

不过本例实际有一个明显的周期性变化。观察图 5-7 可以看到，每年 8 月份的销售峰值均向右移动了，即红色和绿色曲线的销售峰值出现在 9 月份。同时阶数越大，峰值持续时间越长。

注意

简单 / 加权移动平均虽然可以过滤微小的随机变化，突出长期变化趋势。但是如果使用它进行预测，其对指标实际的变化（周期性或长期趋势下的变化）并不敏感，整体上会做相应的平移，而这会导致预测值的滞后性。

5.4　指数平滑

指数平滑预测的数值不仅可以体现指标的长期变化趋势 trend，还能体现其周期性 seasonality 变化趋势。根据平滑次数的不同，指数平滑又分为一次指数平滑、二次指数平滑和三次指数平滑等。其中，一次指数平滑可以建立平滑数据 $s[t]$，并结合使用实际数据 $x[t]$ 对指标 x 进行预测。二次指数平滑通过对平滑数据 $s[t]$ 的差分数据 $s[t]$-$s[t$-1] 进行再次平滑，建立二次平滑数据 $b[t]$，并结合一次平滑的预测值来对指标 x 进行预测，其中 $b[t]$ 可以代表指标 x 的长期趋势。三次指数平滑在二次指数平滑的基础上对 $x[t]$-$s[t]$ 或 $x[t]$ / $s[t]$ 进行三次指数平滑，并建立平滑数据 $p[t]$，然后结合二次平滑的预测值来对指标 x 进行预测，其中 $p[t]$ 可以代表指标 x 的周期性变化趋势。

5.4.1 一次指数平滑

1. 原理

指数平滑的基础公式如下：

$$s[i]=\text{alpha}*x[i]+(1-\text{alpha})*s[i-1] \quad (5\text{-}1)$$

其中 $x[i]$ 是指标的实际数值，$s[i]$ 是一次指数平滑建立的平滑数据。alpha 是平滑系数，其取值范围为 [0,1]，公式 5-1 也可以写为如下形式：

$$s[i]=\text{alpha}*x[i]+(1-\text{alpha})*(x[i-1]+x[i-2]+..+x[1])/(i-1) \quad (5\text{-}2)$$

可见 alpha 越接近 1，平滑后的数据 $s[i]$ 越接近当前时间的数据值 $x[i]$，也就是说，在这种情况下，数据越不平滑；alpha 越接近 0，平滑后的数据 $s[i]$ 越接近前 $i-1$ 个数据的平滑值，即数据越平滑。alpha 的取值标准是：偏差平方的均值（MSE）最小，即各期实际值 $x[i]$ 与平滑数据 $s[i]$ 的差的平方和除以总数据数目最小。

$s[1]$ 等于 alpha*$x[1]$ +（1−alpha）*$s[0]$，其中 $s[0]$ 可以取 $x[1]$ 或者使用几个初始数值的平均值，如 $s[0]=(x[1]+x[2]+x[3])/3$。

一次指数平滑的预测公式如下：

$$s[t+m]=\text{alpha}*x[t]+(1-\text{alpha})*s[t] \quad (5\text{-}3)$$

其中 m 的取值为 1、2、3……。$s[t+m]$ 是预测的指标数据，由公式 5-3 可知，预测数据 $s[t+m]$ 是一个水平直线，和 m 的取值无关。实际上也只使用一次平滑法对无明显变化趋势的指标进行预测。

2. R 语言实现

下面通过自定义函数来对 5.1 节的销售数据进行一次指数平滑分析，并预测 2013 年的全年数据。代码如下：

```
exp1<-function(x,alpha,s0){
    s<-0;s[1]<-alpha*x[1]+(1-alpha)*s0;
    num<-length(x)
    for(i in 2:num){
        s[i]<-alpha*x[i]+(1-alpha)*s[i-1]        #公式5-1
    }
    mse<-mean((s-x)^2)
    x.predict<-alpha*x[num]+(1-alpha)*s[num]     #公式5-3
    out<-list(s=s,mse=mse,predict=x.predict)
    out
}
```

自定义函数 exp1 是一次平滑函数，x 是指标的实际数据，alpha 是平滑系数，s0 是平滑数据的初始数据。输出数据为一个数据框，其 s 列是平滑数据 $s[i]$，mse 列是特定 alpha 取值下的偏差平方的均值（MSE），predict 列是预测数据，由于一次指数平滑预测的数据恒为 alpha*$x[t]$ +（1−alpha）*$s[t]$，因此其预测值只有一个数据。

下面分别把平滑系数 alpha 取值为 0.3、0.5、0.8，通过 exp1 函数计算其偏差平方的均

值（MSE）。

```
exp1(sales.ts,0.3,mean(sales.ts[1:3]))$mse
[1] 97596042
exp1(sales.ts,0.5,mean(sales.ts[1:3]))$mse
[1] 53591958
exp1(sales.ts,0.8,mean(sales.ts[1:3]))$mse
[1] 9940423
```

可以看到，最小值为 53591958，其所对应的 alpha 是 0.5，因此可大致确定 alpha 的取值为 0.5。alpha 为 0.5 时做一次平滑。代码如下：

```
exp1.sol<-exp1(sales.ts,0.3,mean(sales.ts[1:3]))
    plot(c(sales.ts,rep(NA,12)),main="月销售数据：2006年1月--2013年12月",type="b",pch=16,
ylab="月销量（万）",xlab="月份")
    lines(c(exp1.sol$s,rep(exp1.sol$predict,12)), col="red",lwd=2, pch=16)
```

结果展示如图 5-8 所示。

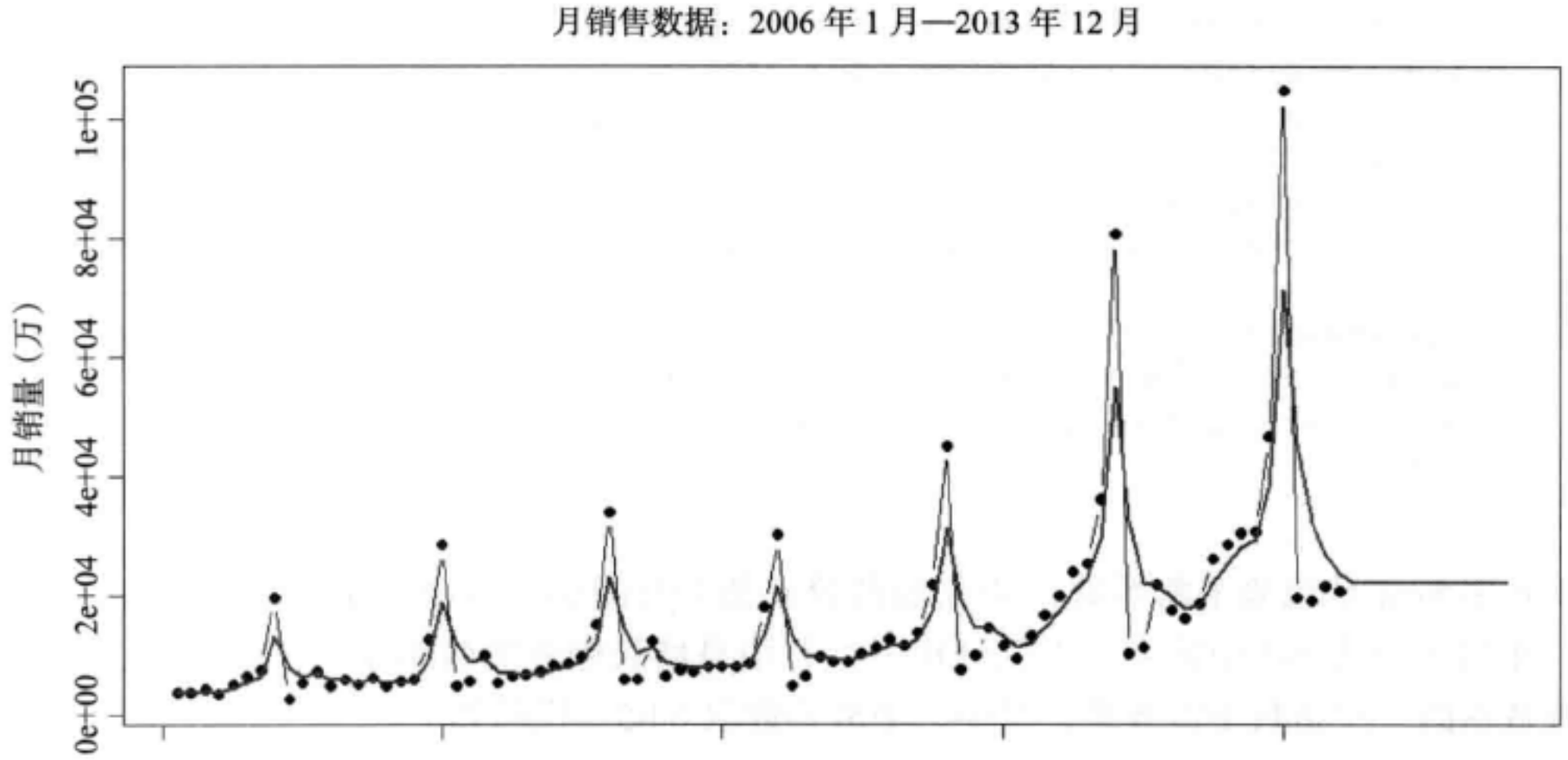


图 5-8 月销量数据和一次指数平滑效果图

其中黑色曲线是实际数据，红色曲线是平滑数据和预测数据。由图 5-8 可见，对于 2013 年 1 月至 2013 年 12 月的数据，一次指数平滑的预测值是一条水平直线，实际上它并未体现出销售指标的长期上升趋势，也未体现出其周期性变化趋势，只能代表 2013 年 1 月的数据。

5.4.2 二次指数平滑

1. 二次指数平滑的原理

一次指数平滑的缺点是，指标所呈现的上升或下降趋势，将导致预测值滞后于真实数

据。通过趋势调整，添加趋势修正值 $b[i]$，可以改进指数平滑预测结果。于此进行改进后，就成为二次指数平滑（霍尔特指数），其公式为 $s[i]$=alpha*$x[i]$ +（1−alpha）*（$s[i-1]$ +$b[i-1]$）。修正值 $b[i]$，可以由平滑数据 $s[i]$ 的差分数据 $s[i]-s[i-1]$ 进行再次平滑得到，其中平滑系数为 beta，称其为趋势平滑系数，$b[i]$ 的二次平滑公式为 $b[i]$=beta*（$s[i]-s[i-1]$）+（1−beta）*$b[i-1]$。最后得到的二次指数平滑的基础公式如下：

$$s[i]=\text{alpha}*x[i]+(1-\text{alpha})*(s[i-1]+b[i-1])$$

$$b[i]=\text{beta}*(s[i]-s[i-1])+(1-\text{beta})*b[i-1] \quad (5\text{-}4)$$

经过长期训练后得到的 b[t] 可以作为每预测一个新时间点时其预测数据的增量，它代表指标的长期变化趋势 trend 的斜率。二次指数平滑的预测公式如下：

$$s[t+m]=s[t]+b[t]*m \quad (5\text{-}5)$$

2. 二次指数平滑的 R 语言实现

下面使用二次指数平滑法自定义一个函数。代码如下：

```
exp.holt<-function(x,alpha,beta,s0,b0,predict.num=1){
    s<-0; s[1]<-alpha*x[1]+(1-alpha)*(s0+b0);
    b<-0; b[1]<-beta*(s[1]-s0)+(1-beta)*b0;
    num<-length(x)
    for(i in 2:num){
        s[i]<-alpha*x[i]+(1-alpha)*(s[i-1]+b[i-1])
        b[i]<-beta*(s[i]-s[i-1])+(1-beta)*b[i-1]          # 公式 5-4
    }
    mse<-mean((s-x)^2)
    x.predict<-s[num]+b[num]*(1:predict.num)              # 公式 5-5
    out<-list(s=s,mse=mse,predict=x.predict)
    out
}
```

其中 beta 是趋势平滑系数，b0 是趋势修正值 b[i] 的初始数据，predict.num 是预测的数目，相当于公式 5-5 中的 m。下面分析一个有明显增长趋势的数据（=3 × t+ 误差），绘制其差分数据的一次指数平滑效果，其中，平滑系数取 0.05。代码如下：

```
    data<-3*c(1:84)+rnorm(84,0,5)
    plot(diff(data),lwd=2, pch=16,type="b")
    lines(exp1(diff(data),0.05,mean(diff(data)[1:3]))$s,lwd=2,pch=16,type="b",
col="red")
```

结果如图 5-9 所示。

在该例中，把 $b[i]$=beta*（$s[i]-s[i-1]$）+（1−beta）*$b[i-1]$ 中的 $s[i]$ 和 $s[i-1]$ 近似使用原始数据 $x[i]$ 和 $x[i-1]$ 代替后，$b[i]$ 就是原始数据 $x[i]$ 差分的一次指数平滑数据。此时可以通过观察图 5-9 中的红色曲线值来预测 $b[i]$。实际上 $b[i]$ 始终在 3 上下随机浮动。即用于代表长期变化趋势的斜率 b[t] 约为 3，这与 data 的实际情况（data 等于 3*t+ 误差）正好相符。最后对 data 数据进行二次指数平滑预测，上例中差分数据的一次指数平滑系数（取值为 0.05）在这里变为了趋势平滑系数 beta，$b0$ 则使用原始数据 data 差分数据（初始 3 个）的平

均数来代替。下面查看最后结果。代码如下：

```
expholt.sol<-exp.holt(data,0.5,0.05,mean(data[1:3]),mean(diff(data[1:4])),12);
plot(c(data,rep(NA,12)),ylim=c(0,400),lwd=2, pch=16,type="b")
lines(c(expholt.sol$s,expholt.sol$predict), col="blue",lwd=2, pch=16)
```

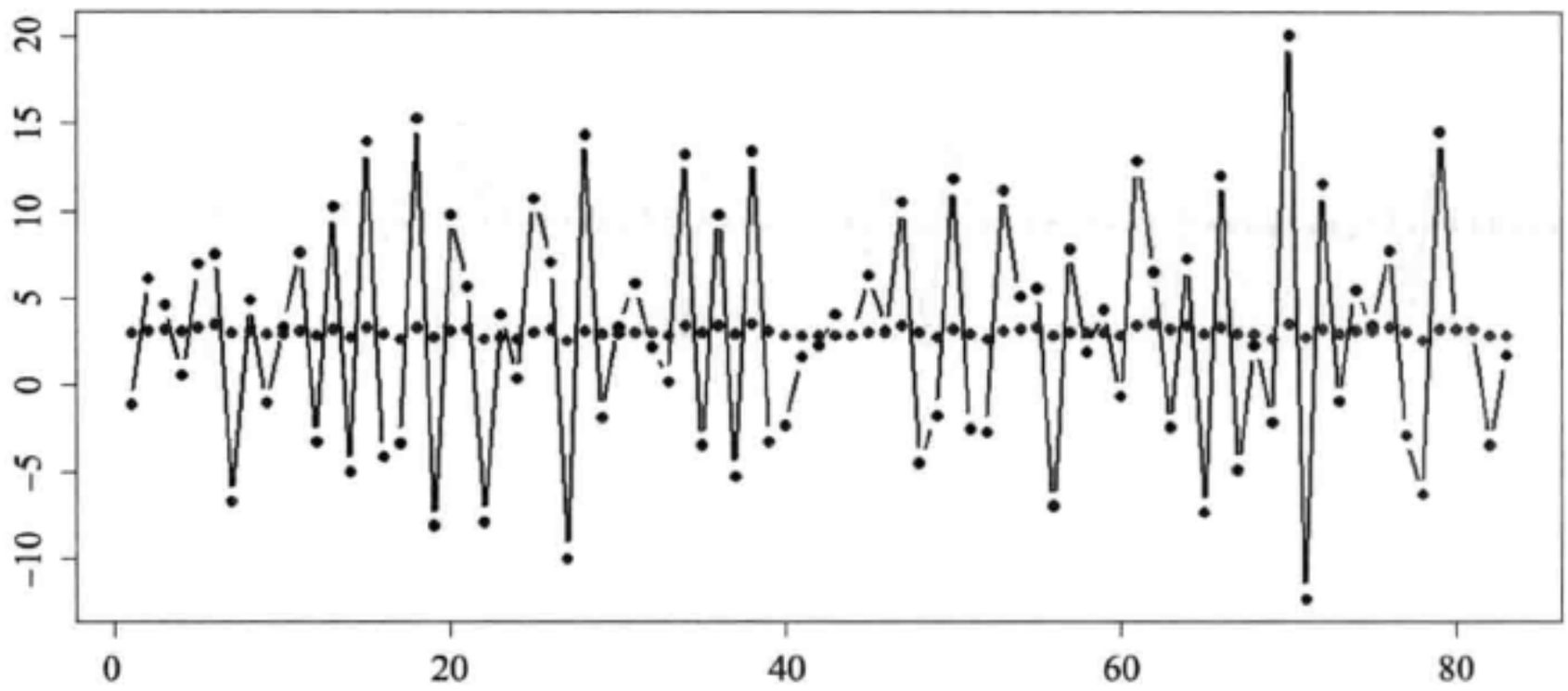


图 5-9　结果展示图

结果如图 5-10 所示。

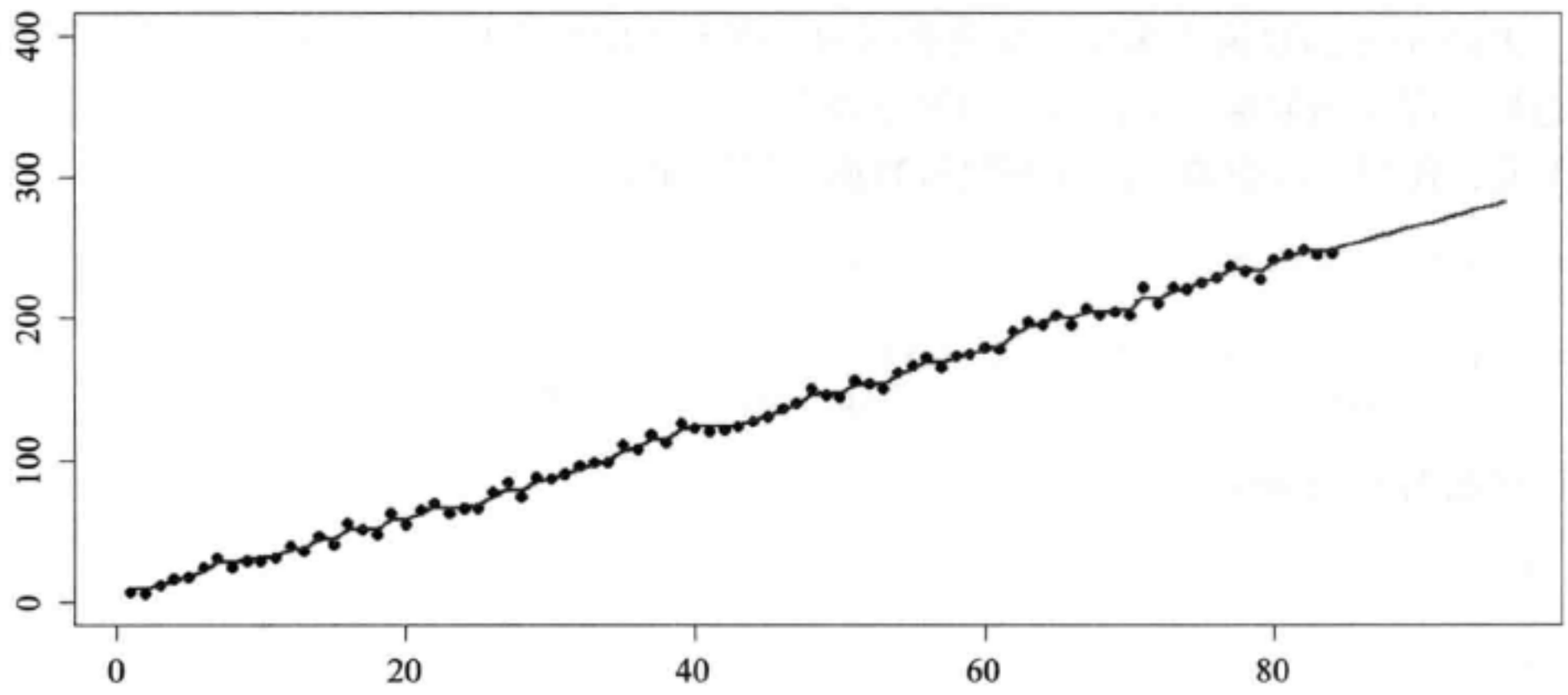


图 5-10　结果展示图

在图 5-10 中，黑色曲线是实际数据，蓝色曲线是二次指数平滑数据和预测数据。由图 5-10 可知，二次指数平滑分析预测数据是一条斜线，其中由二次平滑得到的斜率数据（也是 t 时刻的趋势修正值 $b[t]$）代表了指标的趋势性。

然而对于 5.1 节的销售数据，由于其周期性较大，实际不能使用二次指数平滑进行预测。看看如果使用此法预测会有什么效果，下面是销售数据的差分一次平滑数据。

```
plot(as.numeric(diff(sales.ts)),lwd=2, pch=16,type="b")
```

```
    lines(exp1(diff(sales.ts),0.05,mean(diff(sales.ts)[1:3]))$s,lwd=2,pch=16,type="
b",col="red")
```

结果如图 5-11 所示。

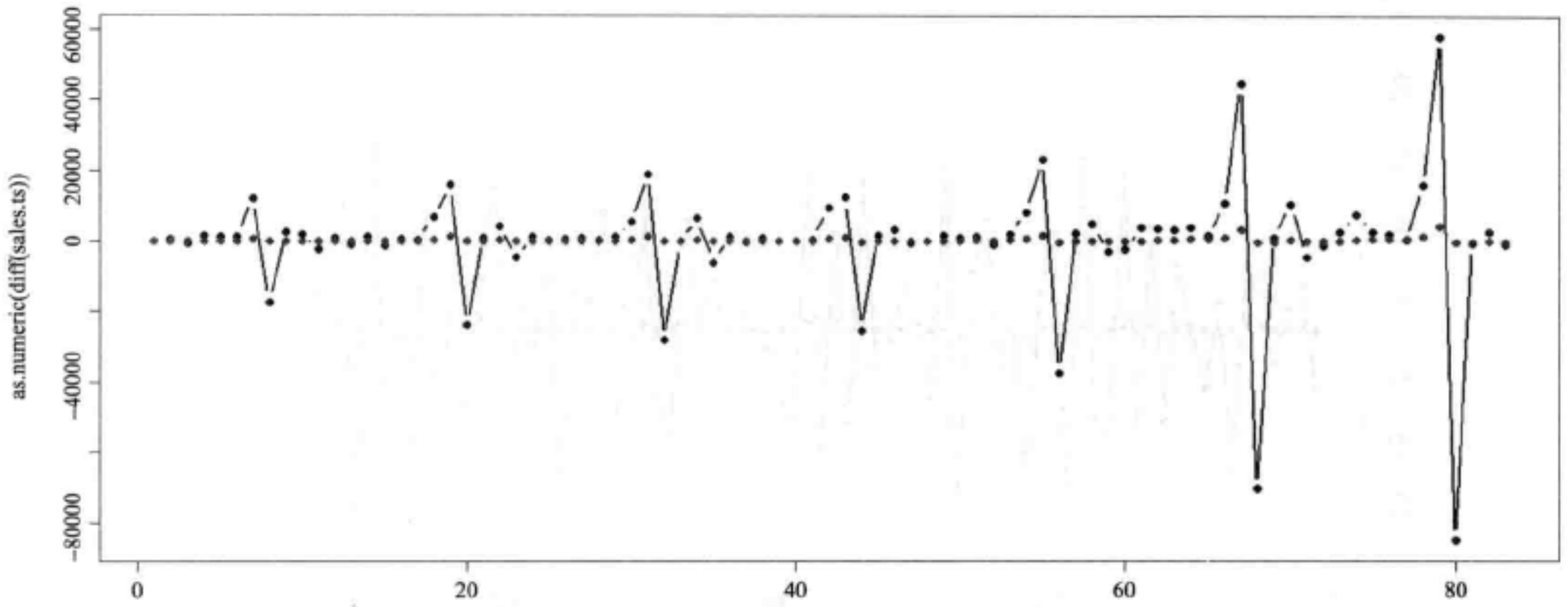


图 5-11 结果展示图

实际数据有明显的周期性变化波动，导致差分数据并非是一个正态分布的随机数据，也就是说，无论设置多小的趋势平滑系数 beta，都很难预测出指标的真实趋势。实际上，在本例中，无论 beta 的取值为多少，预测数据的斜率 $b[t]$ 均小于 0，而实际的销售数据明显有上升的趋势，即预测数据的拟合直线斜率应该大于 0。

下面，使用二次指数平滑预测销售数据。代码如下：

```
    expholt.sol<-exp.holt(sales.ts,0.5,0.05,mean(sales.ts[1:3]),mean(diff(sales.
ts[1:4])),12);
    plot(c(sales.ts,rep(NA,12)),lwd=2, pch=16,type="b")
    lines(c(expholt.sol$s,expholt.sol$predict), col="blue",lwd=2, pch=16)
```

结果如图 5-12 所示。

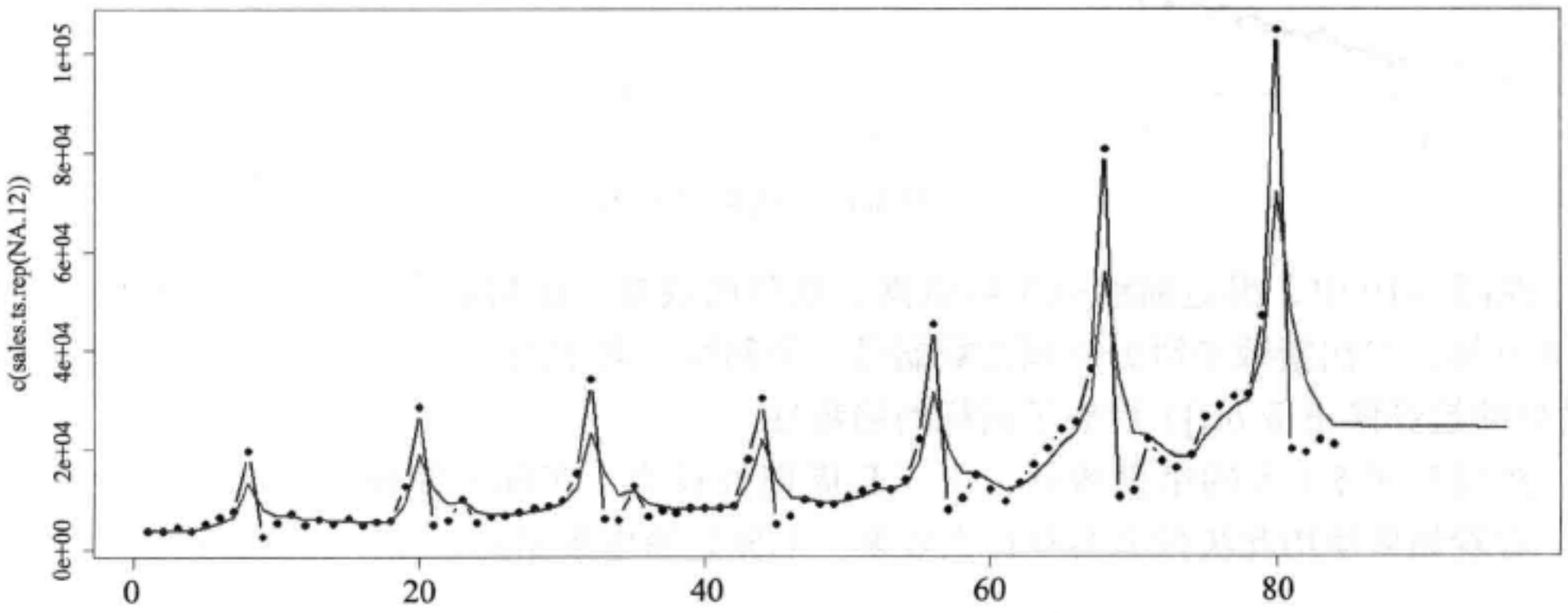


图 5-12 月销量数据和二次指数平滑效果图

在图 5-12 中，最右侧的蓝色曲线是 2013 年 12 个月的预测数据，大致呈现出水平直线，这是明显不正确的。那么，这种情况下，应该使用哪种方式预测呢？实际上在考虑周期变化时，应该使用三次指数平滑法。

上面自定义了二次指数平滑函数，但是它并未给出如何选择两个平滑系数 alpha 和 beta，在 R 语言中，可以使用 holtwinters 函数来找到适当的平滑系数，并给出平滑后的模型进行预测，在三次指数平滑法中会使用到该函数。

5.4.3 三次指数平滑

1. 三次指数平滑的原理

三次指数平滑（温斯特指数平滑）依据预测公式可以分为累加和累乘两种形式。

（1）累加形式

如果在二次指数平滑的基础上对其添加周期性变化趋势修正值 $p[i]$，则成为三次指数平滑，其公式为 $s[i]=\text{alpha}*(x[i]-p[i-L])+(1-\text{alpha})*(s[i-1]+b[i-1])$，其中 L 是周期长度。修正值 $p[i]$ 可以由实际数据 $x[i]$ 和平滑数据 $s[i]$ 的差值 $x[i]-s[i]$ 进行三次平滑得到，其中平滑系数为 gamma，我们称之为周期平滑系数，$p[i]$ 的三次平滑公式为 $p[i]=\text{gamma}*(x[i]-s[i])+(1-\text{gamma})*p[i-L]$。注意，这里使用的是 $p[i-L]$，而非 $p[i-1]$，即 $p[i]$ 具有长度为 L 的周期性特征。最后得到三次指数平滑的基础公式如下：

$$\begin{aligned} s[i]&=\text{alpha}*(x[i]-p[i-L])+(1-\text{alpha})*(s[i-1]+b[i-1]) \\ b[i]&=\text{beta}*(s[i]-s[i-1])+(1-\text{beta})*b[i-1] \\ p[i]&=\text{gamma}*(x[i]-s[i])+(1-\text{gamma})*p[i-L] \end{aligned} \tag{5-6}$$

三次指数平滑的预测公式如下：

$$s[t+m]=s[t]+b[t]*m+p[t-L+(m \bmod L)] \tag{5-7}$$

（2）累乘形式

与累加形式类似，可以由实际数据 $x[i]$ 和平滑数据 $s[i]$ 的比值 $x[i]/s[i]$ 进行三次平滑得到周期性变化趋势修正值 $p[i]$，并得到三次指数平滑的基础公式如下：

$$\begin{aligned} s[i]&=\text{alpha}*(x[i]/p[i-L])+(1-\text{alpha})*(s[i-1]+b[i-1]) \\ b[i]&=\text{beta}*(s[i]-s[i-1])+(1-\text{beta})*b[i-1] \\ p[i]&=\text{gamma}*(x[i]/s[i])+(1-\text{gamma})*p[i-L] \end{aligned} \tag{5-8}$$

三次指数平滑的预测公式如下：

$$s[t+m]=(s[t]+b[t]*m)*p[t-L+(m \bmod L)] \tag{5-9}$$

2. HoltWinters 函数

下面介绍如何使用 HoltWinters 函数实现二次指数和三次指数平滑分析。该函数的具体参数如表 5-3 所示。

表 5-3 HoltWinters 函数的参数

参数	说明
x	时间序列指标的实际数据
alpha	一次平滑系数
beta	二次（趋势）平滑系数，如果取值为 FALES，则 HoltWinters 函数实现一次指数平滑，否则设置为 [0,1]
gamma	三次（周期）平滑系数，如果取值为 FALES，则 HoltWinters 函数不实现三次指数平滑（不预测周期性指标），否则设置为 [0,1]
seasonal	三次指数平滑的方式：取值为 additive 表示使用累加方式，取值为 multiplicative 表示使用累乘方式。如果 gamma 取值为 FALSE，则参数无效
start.periods	计算周期长度（相当于三次指数平滑公式中的 L）前的初始值。如果 gamma 取值为 FALSE，则参数无效
l.start	相当于指数平滑公式中的初始值 $s[0]$
b.start	相当于二次指数或三次指数平滑公式中的初始值 $b[0]$。如果 beta 取值为 FALSE，则参数无效
s.start	相当于三次指数平滑公式中的初始值 $p[1\text{-}L]...p[-1]$，$p[0]$。如果 gamma 取值为 FALSE，则参数无效
optim.start	依据偏差平方的均值（MSE）最小原则计算出的最佳平滑参数 alpha、beta、gamma 前的初始值

仍使用 5.1 节中的销售数据来进行说明。代码如下：

```
sol<-HoltWinters(sales.ts);sol
Smoothing parameters:
 alpha:  0.02456435
 beta :  1
 gamma:  1
Coefficients:
            [,1]
a    34408.8595
b      449.0218
s1  -11664.6955
s2   -9787.9010
s3   -2928.7312
<中间未展示>
s11 -12363.2177
s12 -13797.6895
plot(sol)
```

上面使用 HoltWinters 函数计算出最佳平滑系数 alpha 等于 0.02456435，beta 等于 1，gamma 等于 1。在预测模型系数结果 Coefficients 中，a 是公式 5-9 中的 $s[t]$, b 是 $b[t]$, 同时计算出销售数据的周期长度为 12，系数为 $s1 \sim s12$。最后结果如图 5-13 所示，使用二次指数平滑模拟预测的时间序列模型仍对实际数据的趋势周期性变化很敏感。

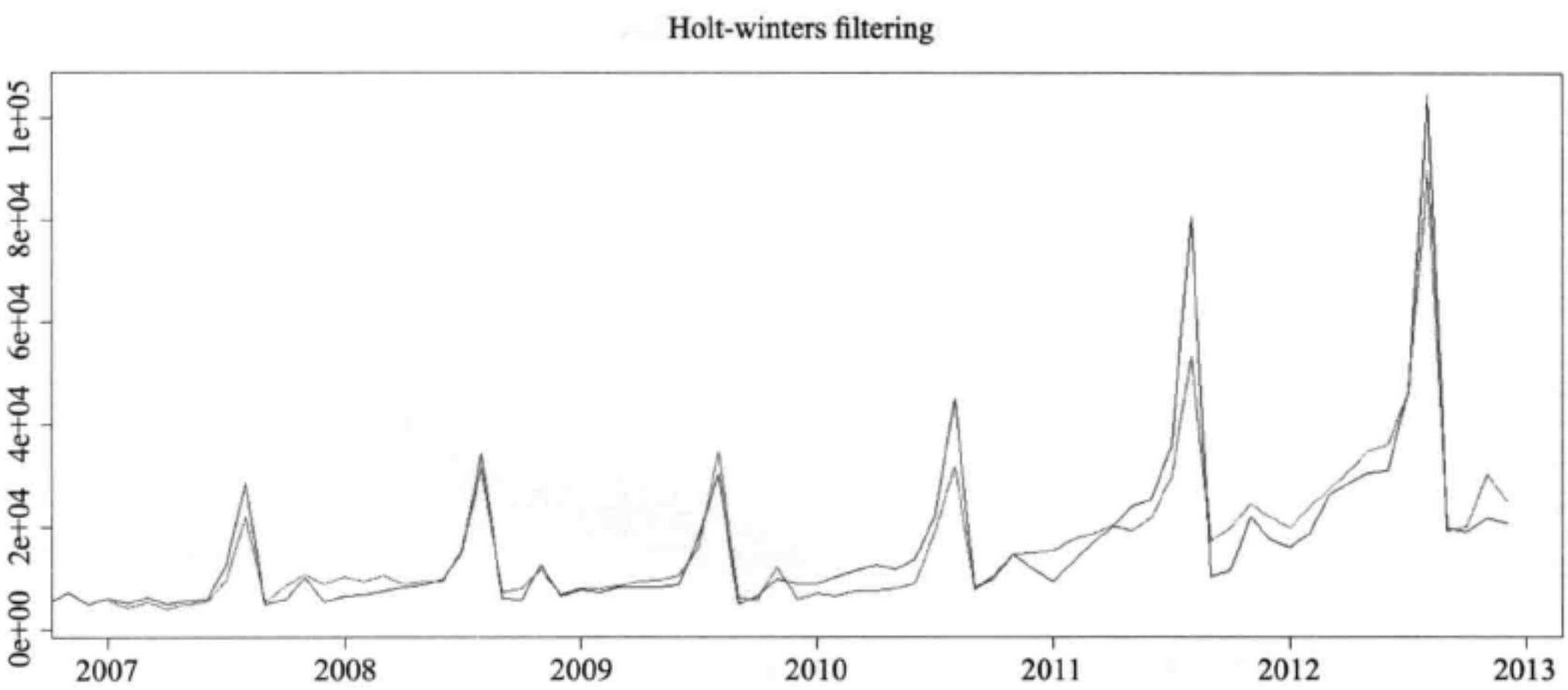


图 5-13 月销量数据和三次指数平滑效果图

使用如下函数可以预测 HoltWinters 函数产生的模型的新时间序列。

```
library("forecast")
sol.forst<-forecast.HoltWinters(sol, h=12);sol.forst
          Point Forecast     Lo 80     Hi 80     Lo 95     Hi 95
Jan 2013       23193.19  16567.62  29818.75  13060.26  33326.12
Feb 2013       25519.00  18885.44  32152.56  15373.85  35664.15
Mar 2013       32827.19  26175.69  39478.70  22654.60  42999.79
<中间省略>
Nov 2013       26984.88  19416.87  34552.89  15410.61  38559.16
Dec 2013       25999.43  18183.48  33815.39  14045.96  37952.90
plot.forecast(sol.forst,include=10)#绘制预测曲线，包含预测数值前 10 个原始数据
```

其中，forecast.HoltWinters 函数用于对 HoltWinters 函数产生的模型进行预测，参数中的 h=12 表明对未来 12 个月进行预测。在展示的结果中，Forecast 列是预测值，也可以使用 sol.forst$mean 来直接调用。Lo 80 和 Hi 80 两列分别是在 80% 置信度情况下预测的置信范围，可以直接使用 sol.forst$lower[,1] 和 sol.forst$upper[,1] 调用。Lo 95 和 Hi 95 两列分别是在 95% 置信度情况下预测的置信范围，可以直接使用 sol.forst$lower[,2] 和 sol.forst$upper[,2] 调用。也可以通过设置 forecast.HoltWinters 函数中的 level 参数（默认为 c（80,95）取值）来改变置信度，进而计算其预测范围。这里有必要理解一下置信度和置信区间的概念。置信度可以理解为可信度，置信区间则可以理解为在满足该可能性时预测数据的范围。例如，上例中在 80% 和 95% 置信度下，2013 年 1 月的置信区间为 [16567.62，29818.75] 和 [13060.26，33326.12]，那么就可以理解为预测 2013 年 1 月的销售数据时，有 80% 的可能性会处于 [16567.62，29818.75] 范围内，有 95% 的可能性会处于 [13060.26，33326.12] 范围内，如图 5-14 所示。可见，置信度越高，预测的取值范围就越大（置信范围越大）。plot.forecast 函数用于绘制预测图，其中 include=10 表明绘制预测前 10 个月的数据和未来 12 个月的预测数据。

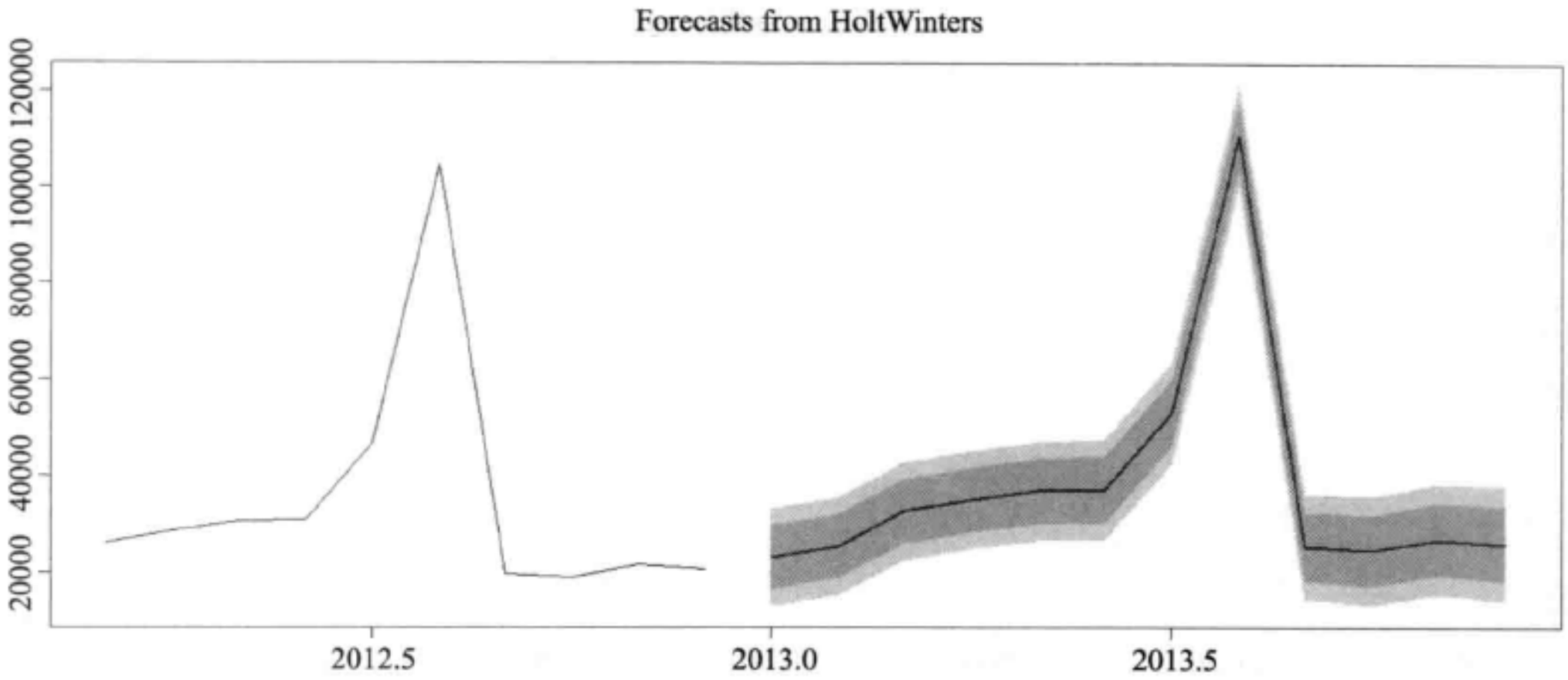


图 5-14 三次指数平滑的预测及区间图

在图 5-14 中，深蓝色的曲线是预测数值，浅蓝色区域是 80% 置信度下的置信区间，灰白色区域是 95% 置信度下的置信区间。

5.5 ARIMA 模型

ARIMA 模型是时下较为流行的时间序列分析模型，它通过对白噪声的组合来建立模型，并保证模型和实际数据的残差也是一个均值很小的正态分布数据。ARIMA 模型的处理场景如表 5-4 所示。

表 5-4 ARIMA 模型的处理场景

序列组成	类型	模型
误差	平稳序列	MA（q）;AR（p）;ARMA（p,q）
长期变化趋势 + 误差	无周期性的非平稳序列	ARIMA（p,d,q）
周期性变化趋势 + 长期变化趋势 + 误差	有周期性的非平稳序列	周期性 ARIMA（p,d,q）

5.5.1 自相关性

1. 协方差矩阵和相关系数

协方差矩阵和相关系数主要用于研究两个连续型变量向量的相似程度（相关性）。例如，变量 x 的 n 个样本为 $x1$，$x2$，$x3$，$x4$，…，xn。变量 y 的 n 个样本为 $y1$，$y2$，$y3$，$y4$，…，yn，则有：

变量 x 和变量 y 的样本均值分别为：

$$\bar{x}=\frac{1}{n}\sum_{i=1}^{n}x_i,\quad \bar{y}=\frac{1}{n}\sum_{i=1}^{n}y_i$$

变量 x 的样本方差为：

$$s_{xx}=\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2$$

变量 y 的样本方差为：

$$s_{yy}=\frac{1}{n-1}\sum_{i=1}^{n}(y_i-\bar{y})^2$$

变量 x 和 y 的样本协方差为：

$$s_{xy}=\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})$$

变量 x 和 y 的协方差矩阵为：

$$S=\begin{bmatrix}s_{xx} & s_{xy}\\ s_{xy} & s_{yy}\end{bmatrix}$$

变量 x 和 y 的样本相关系数为：

$$r_{xy}=\frac{s_{xy}}{\sqrt{s_{xx}}\sqrt{s_{yy}}}$$

变量 x 和 y 的相关系数矩阵为：

$$R=\begin{bmatrix}r_{xx} & r_{xy}\\ r_{xy} & r_{yy}\end{bmatrix}=\begin{bmatrix}1 & r_{xy}\\ r_{xy} & 1\end{bmatrix}$$

在 R 语言中，可以使用以数据框形式输入的 cov（data.frame（x, y））语句来计算协方差矩阵 S，或者使用以向量形式输入的 cov（x, y）语句直接计算变量 x 和 y 的样本协方差 s_{xy}。使用 cor（data.frame（x, y））计算相关系数矩阵 R，使用 cor（x, y）计算变量 x 和 y 的样本相关系数 r_{xy}，并使用它来衡量变量 x 和 y 的相似程度。例如：

```
par(mfrow=c(1,3))
data1<-data.frame(x=rnorm(100,20,1),y=rnorm(100,200,10))
plot(data1,main=paste("相关系数=", cor(data1)[[2]]))
x<- rnorm(100,20,1)
y<-29*x+ rnorm(100,10,10)
data2<-data.frame(x,y)
plot(data2,main=paste("相关系数=", cor(data2)[[2]]))
x<- rnorm(100,20,1)
y<- -19*x+ rnorm(100,10,10)
data3<-data.frame(x,y)
plot(data3,main=paste("相关系数=", cor(data3)[[2]]))
```

结果如图 5-15 所示。

在图 5-15 中，左边散点图中的变量 x 和 y 并无明显联系，变量 y 很难使用变量 x 来表达（实际上 y 是均值为 200，标准差为 10 的正态分布样本点，与 x 无任何关系）。此时计算的相关系数 cor（data1）[[2]] 等于 -0.162。

在中间的散点图中，随着变量 x 取值的增大，变量 y 有明显的增加趋势（实际上 y 是由变量 x 乘以常数 29 再加上微小误差 rnorm（100,10,10）模拟而成的）。此时计算的相关系数 cor（data2）[[2]] 等于 0.955，很接近于极限值 1，这时称变量 x 和 y 是正相关的。

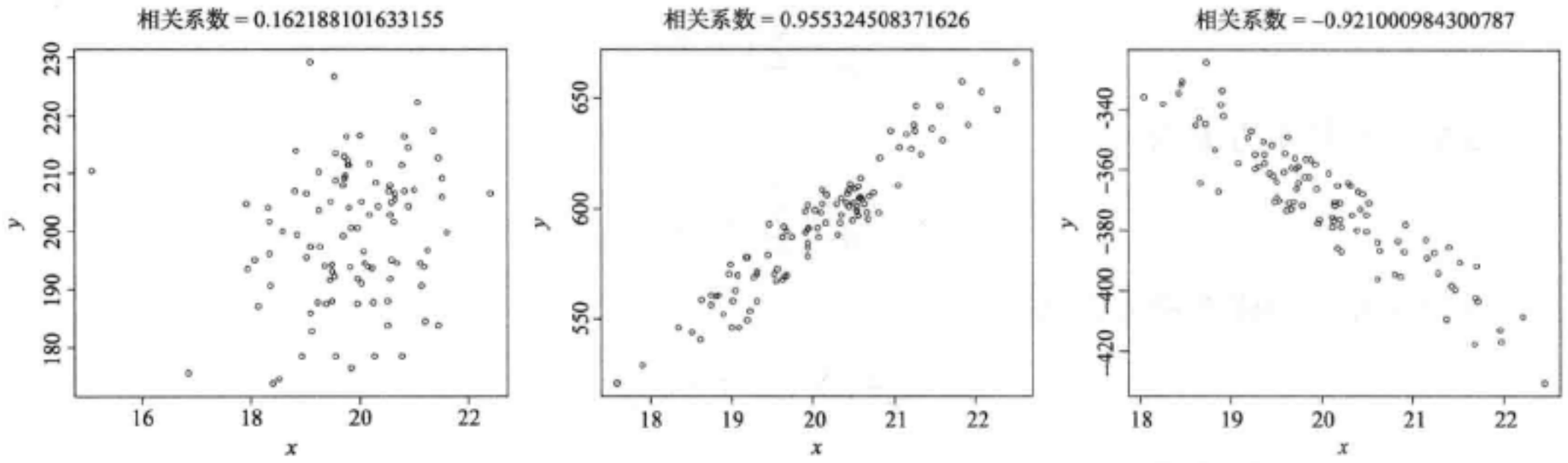


图 5-15 相关系数实例

在右侧的散点图中，随着变量 x 取值的增大，变量 y 有明显的减小趋势（实际上 y 是由变量 x 乘以常数 -19 再加上微小误差 rnorm（100,10,10）模拟而成的）。此时计算的相关系数 cor(data3)[[2]] 等于 -0.921，很接近于极限值 -1，这时称变量 x 和 y 是负相关的。

协方差和相关系数会在第 6 章详细介绍，这里只需理解相关系数的含义即可。

- 在 x 增大的同时，如果 y 也随之增大的趋势越明显（y 可以近似写为 $y=ax+b,a>0$），则相关系数 r_{xy} 越接于近极限值 1。
- 在 x 增大的同时，如果 y 随之减小的趋势越明显（y 可以近似写为 $y=ax+b,a<0$），则相关系数 r_{xy} 越接近于极限值 -1。
- x 和 y 的关系越不明显（y 无法使用 x 表示），则相关系数 r_{xy} 越接近于 0。

2. 自相关性和自相关图

假设有一个序列记为 $Xt=c(x1,x2,x3,\cdots,xn)$，则该序列的 1 阶滞后（lag=1）序列记为 $Xt\text{-}1=c(x2,x3,x4,\cdots,xn)$，同理，k 阶滞后（lag=k）序列记为 $Xt\text{-}k=c(xk+1,xk+2,xk+3,\cdots,xn)$。而序列 Xt 的自相关性就是原始序列 Xt 与各阶滞后序列 $Xt\text{-}k$ 之间的相关性。通常认为如果序列 Xt 和 $Xt\text{-}k$ 的相关系数处于 [-0.2,0.2] 范围内，则无相关性。

使用 arima.sim 函数模拟 100 个由 AR 过程（会在 5.5.4 节详细讲解）产生的序列。代码如下：

```
data<-arima.sim(n=100,list(ar=c(0.8,-0.1)))
lag.plot(data , 16, do.lines=FALSE)
```

结果如图 5-16 所示。

lag.plot 函数可以分别绘制原始数据 data 与 k 阶滞后（lag=k）的散点图，其中 y 轴是原始数据 data，x 轴是 k 阶滞后数据。通过观察各个子图可知，data 序列同 1 阶滞后的序列相关性较强，3 阶以后几乎无任何相关性可言。接下来计算各阶滞后与原序列的自相关系数 cor.lag。代码如下：

```
cor.lag=0
for(k in 1:16){
     cor.lag[k]<-cor(data[-c((101-k):100)],data[-c(1:k)])
```

```
}
cor.lag
 [1]  0.605184107   0.312766432   0.126671372   0.024450397
 [5]  0.007814811   0.039322999   0.058460693   0.049194781
 [9]  0.019117471  -0.082039621  -0.183846163  -0.113096956
[13] -0.094045192  -0.138607095  -0.038637102  -0.074261730
```

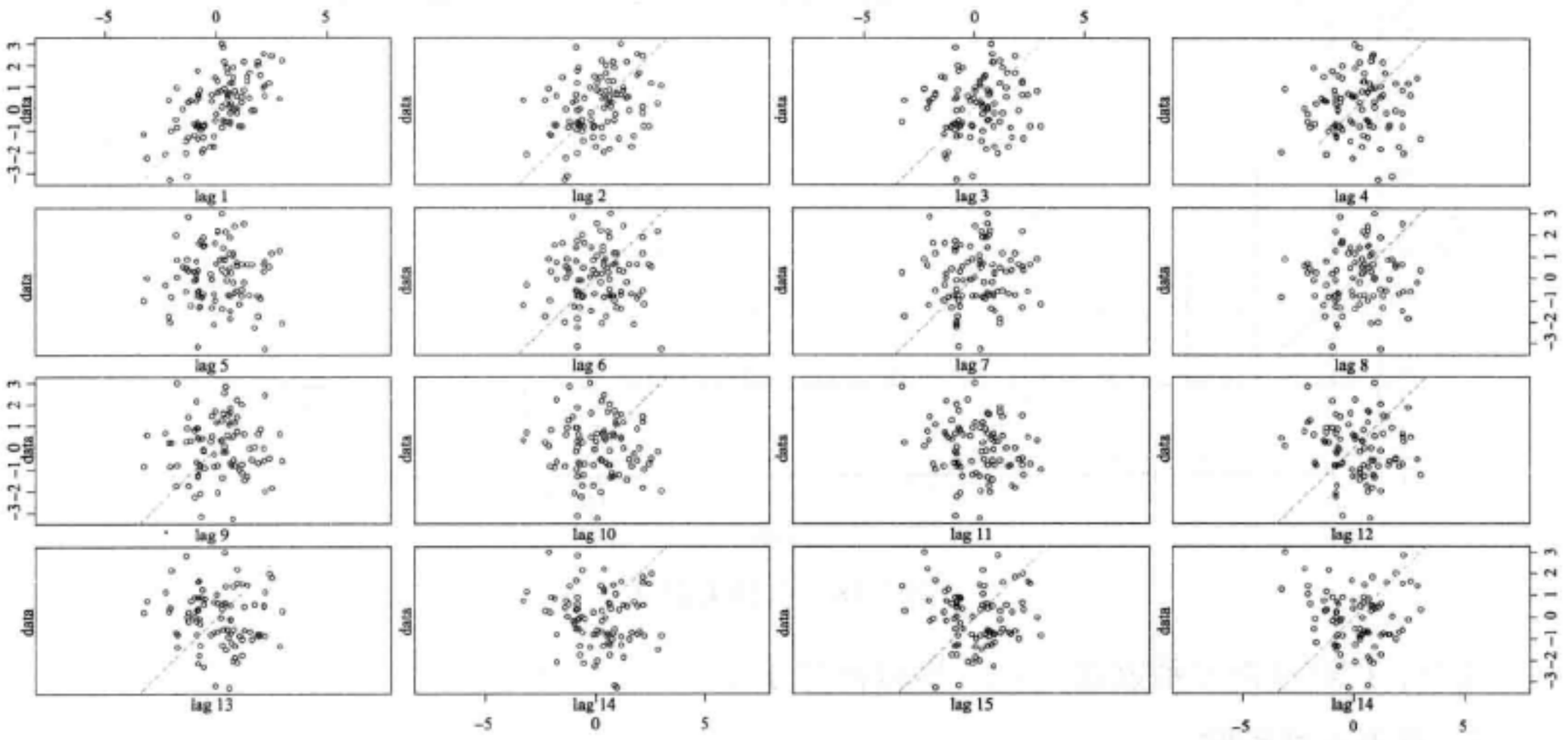


图 5-16　k 阶滞后散点图

然后把 cor.lag 绘制成图形，并在 y 轴等于 −0.2 和 0.2 时绘制横向虚线。

```
plot(cor.lag,type= "h"); abline(h=0);
abline(h=-0.2,lty= "dashed",col="red");
abline(h=0.2,lty= "dashed",col="red");
```

结果如图 5-17 所示。

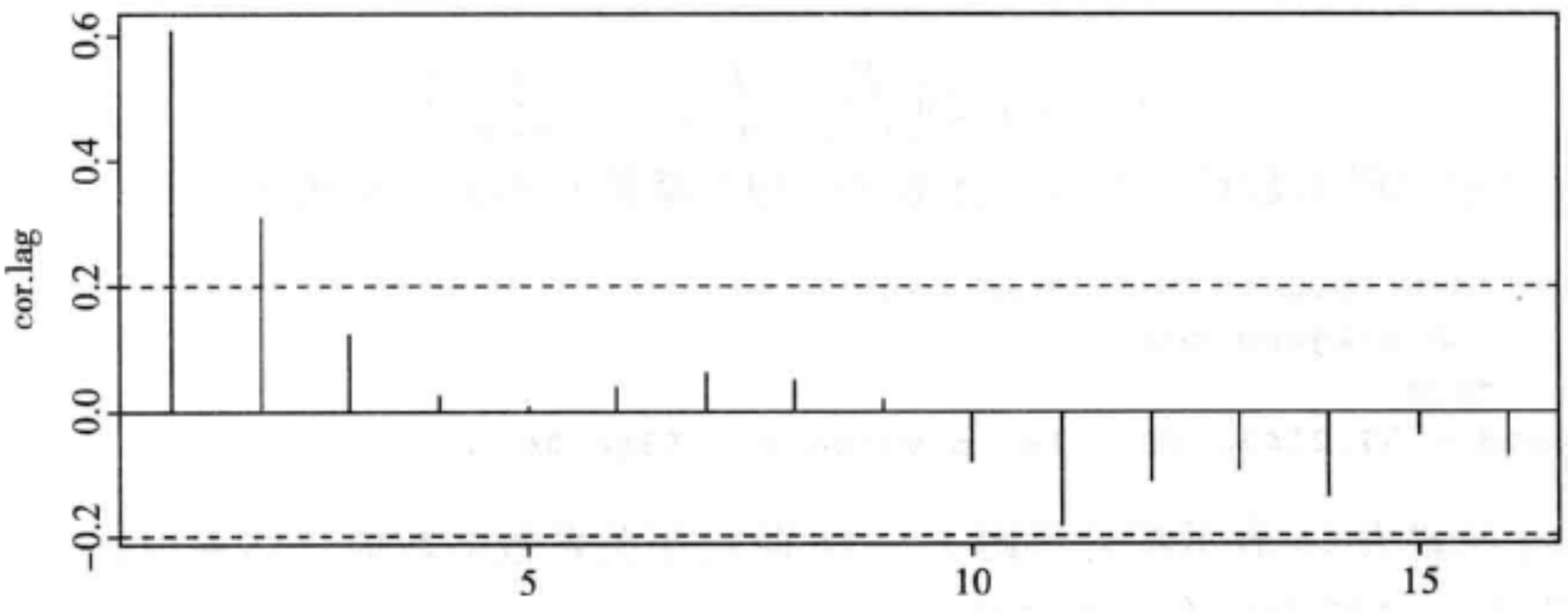


图 5-17　自相关图 1

图 5-17 也可以说明原序列 data 与滞后 3 阶后序列之间无自相关性。该图为自相关图，

在 R 语言中可以使用 acf 函数快速计算并绘制出自相关图。下面绘制 0 ～ 16 阶的自相关图。代码如下：

```
acf(data,lag.max=16)
```

结果如图 5-18 所示。

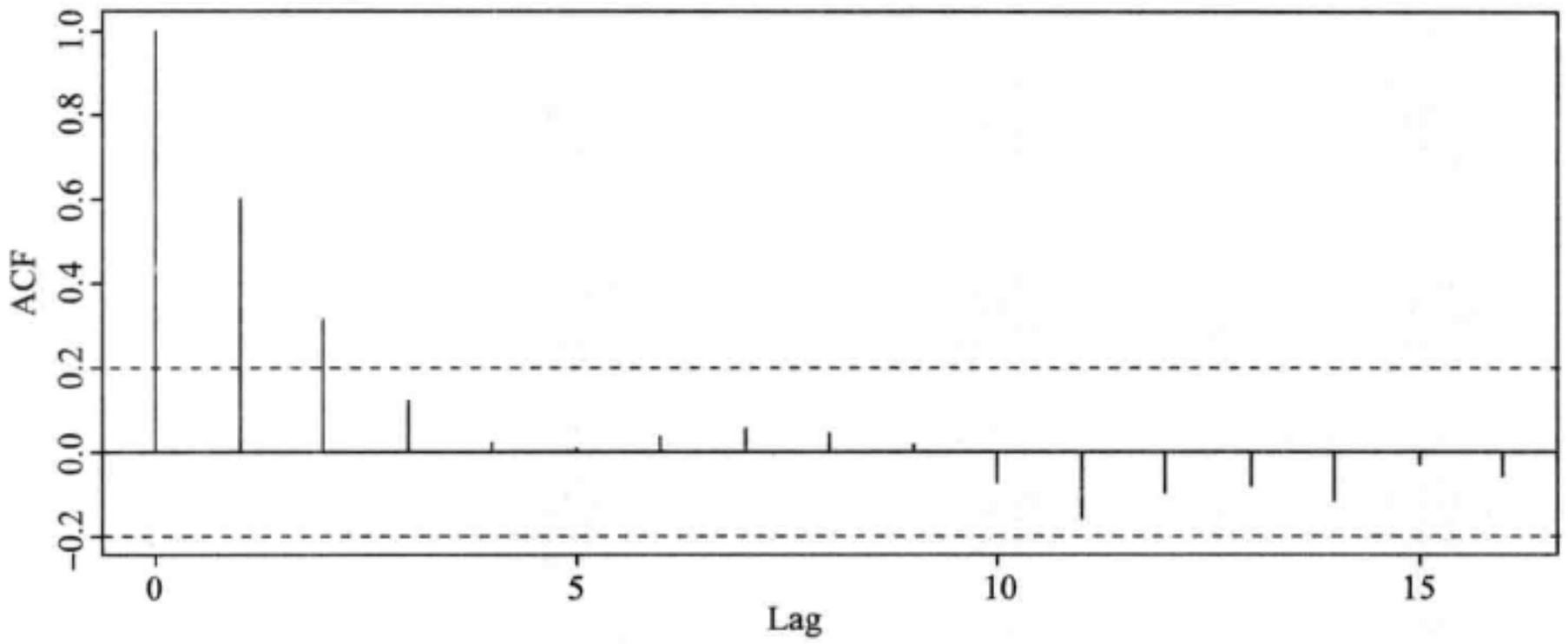


图 5-18 自相关图 2

注意，0 阶自相关系数就是 r_{xx}，所以恒等于 1。

3. 自相关性检验

如果有任何一阶滞后序列与原始序列 *Xt* 的相关系数不处于 [0.2，−0.2] 范围内，则称序列 *Xt* 不具有自相关性。这时可以通过 acf 函数来绘制序列的自相关图，从而观察原始序列与各阶滞后序列的自相关系数值。然而不能简单地认为：只要所有阶数的滞后序列与原始序列的相关系数均处于 [0.2，−0.2] 范围内，序列 *Xt* 就无自相关性。因为有些相关系数可能与临界点 ±0.2 很接近，而多次累计后，作为整体，序列 *Xt* 仍可能具有自相关性。此时，可以使用 Ljung-Box 方法进行检验，对各阶滞后的相关系数进行综合考量，并建立出统计量。代码如下：

$$Q = n(n+2)\left(\frac{p_1^2}{n-1} + \frac{p_2^2}{n-2} + \cdots + \frac{p_K^2}{n-K}\right)$$

其中 *n* 是序列样本数目，*K* 是综合考量的相关系数。相关语句如下：

```
Box.test(data,type="Ljung-Box",lag=160)
        Box-Ljung test
data:   data
X-squared = 57.2146, df = 16, p-value = 1.531e-06
```

其中 lag 设置为 16 表示综合考虑 1 ～ 16 阶自相关系数的数值，结果为 p-value=1.531e-06<0.05，则表示序列 data 有自相关性。

4. 偏自相关图

在自相关性的研究中，*Xt* 和 *k* 阶滞后序列 *Xt-k* 的相关性实际受到 *Xt*-1.... *Xt-k*+1 的影响。

而偏自相关图就是在消除 *Xt*-1... *Xt-k*+1 的影响后，研究 *Xt* 和 *Xt-k* 的相关性。在 *R* 语言中，使用 pacf 函数来绘制偏自相关图。代码如下：

```
pacf(data,lag.max=16)
```

结果如图 5-19 所示。

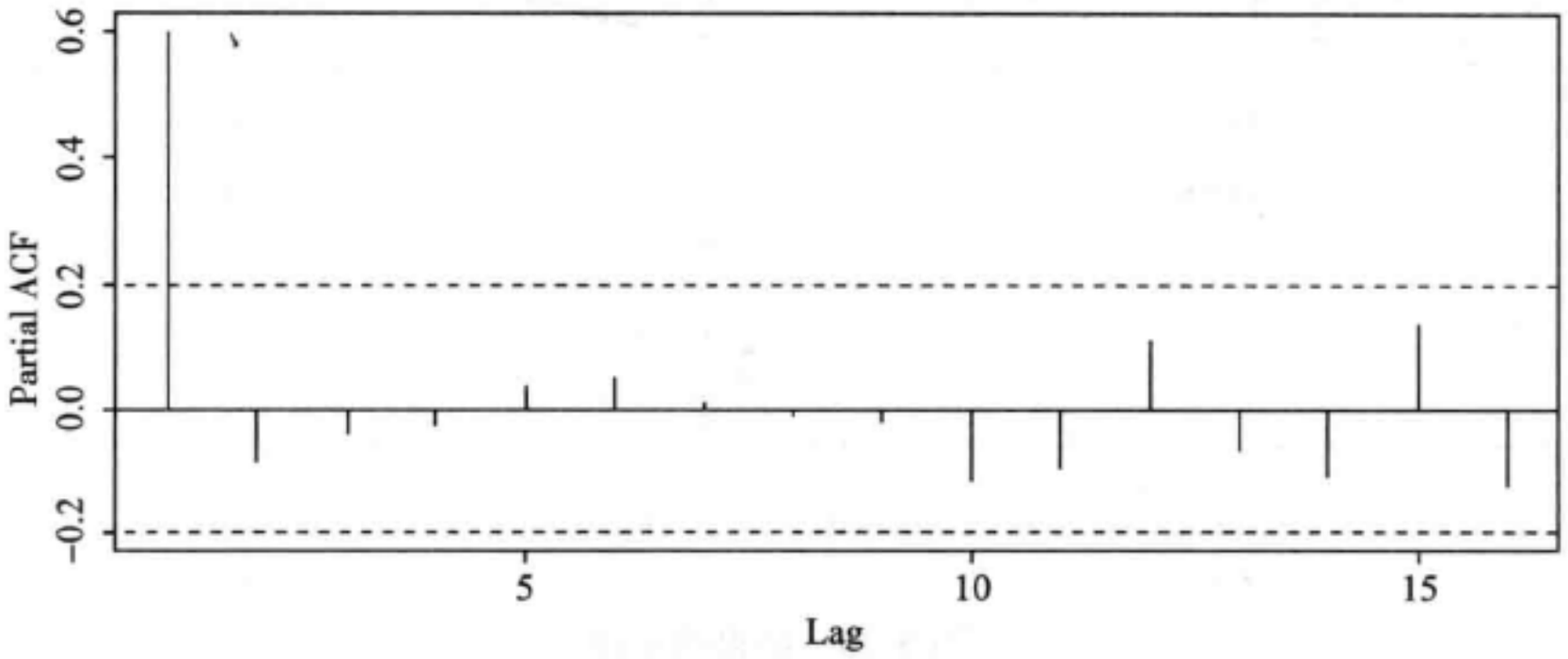


图 5-19 偏自相关图

注意，lag=0 时，无偏自相关系数。对于 pacf 图的应用会在后续章节中说明。

5.5.2 平稳性和白噪声

1. 平稳性

如果某一时间序列是由某一随机过程生成的，即时间序列 *x*(*t*)(*t*=1，2，3…) 的每一个数值都是从同一个概率分布中随机得到的，经由该随机过程所生成的时间序列满足下列条件：

- 均值 *E*(*Xt*) 是与时间 *t* 无关的常数，即任何 *k* 阶滞后序列 *Xt-k* 的均值均相同。
- 方差 Var(*Xt*) 是与时间 *t* 无关的常数，即任何 *k* 阶滞后序列 *Xt-k* 的方差均相同。
- 协方差 Cov(*Xt*,*Xt-k*) 只与滞后阶数 *k* 有关，与 *t* 无关。由于任何 *k* 阶滞后的方差均相同，所以此条件也可以理解为：任何 *k* 阶滞后序列 *Xt-k* 与 *Xt* 的自相关系数只与滞后阶数 *k* 有关，与 *t* 无关。

则称经由该随机过程生成的时间序列是平稳的。平稳性可以通过查看 acf 自相关图来确定，如果 acf 成指数级衰减，则表示是平稳的。一个服从正态分布的随机序列就是一个平稳序列，例如：

```
data1<-rnorm(100,10,1)
data2<-(1:100)*rnorm(100,10,1)
par(mfcol=c(2,2))
plot(data1,main=("平稳序列 - 散点图"),pch=16,type="b")
acf(data1,20)
plot(data2,main=("非平稳序列 - 散点图"),pch=16,type="b")
acf(data2,20)
```

最后的结果如图 5-20 所示。

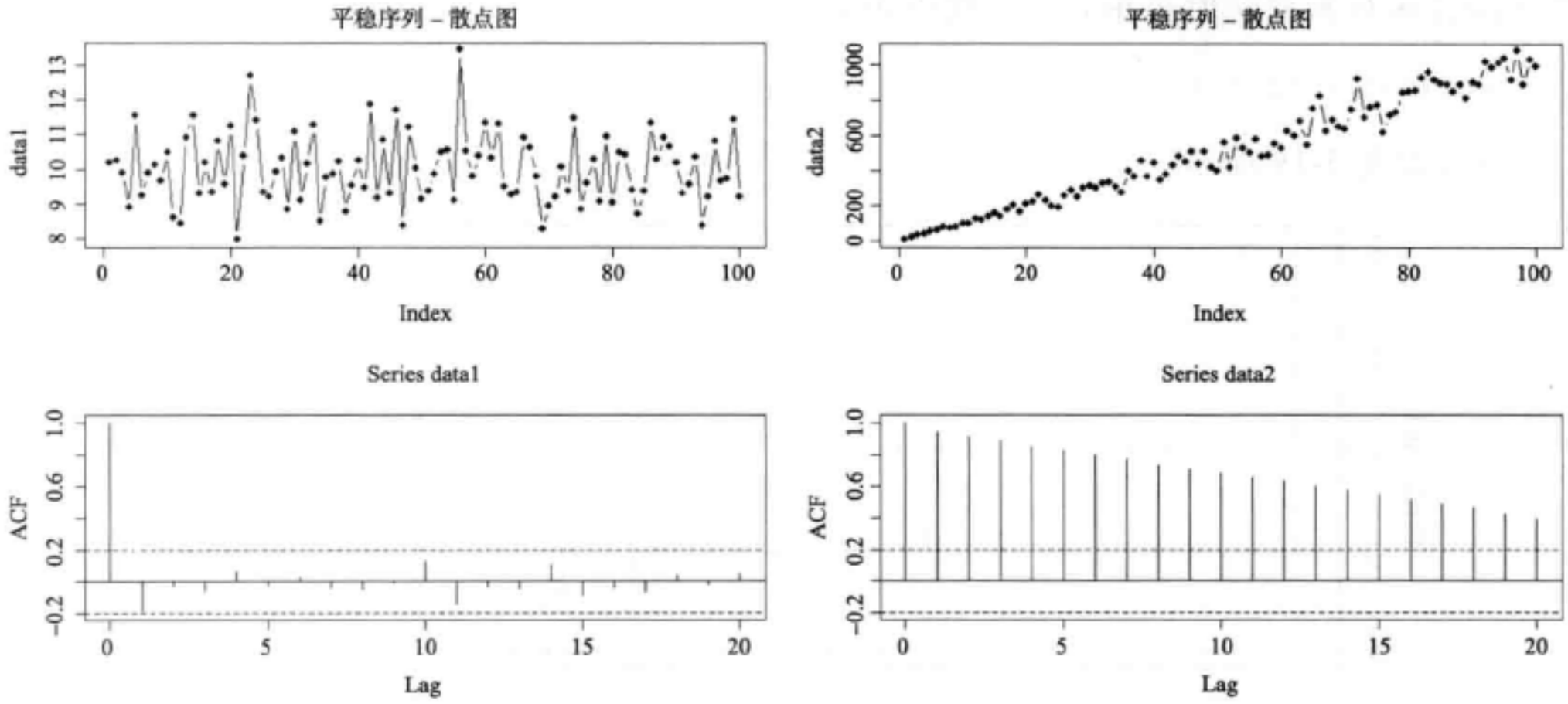


图 5-20　结果展示图

2. 白噪声

对于一个平稳性的时间序列 e_t，如果其均值 $E(e_t)$ 等于常数 0，且无自相关性，则称此平稳性时间序列为白噪声序列。白噪声本身并没有任何研究的意义，其意义在于后续的 ARMA 模型中均需要使用它构造出来。标准正态分布以及均匀分布可以模拟出一个白噪声序列 e_t。相关语句如下：

```
et<-rnorm(50)
et<-runif(50)
```

5.5.3　MA 滑动平均过程

1. 模型

滑动平均过程是通过把若干白噪声做加权（系数为 1,θ1, θ2,…, θ_q）平均得到的，其公式如下：

$$Y_t = e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q} \tag{5-10}$$

上述公式为 q 阶滑动平均过程，记为 MA(q)。Y_{t+1} 由 e_{t+1}, e_t, e_{t-1},…, e_{t-q+1} 的加权平均得到，并以此类推滑动至 Y_{t+2}, Y_{t+3}…。

2. 相关图特征以及阶数 q 的确定

下面使用 arima.sim 函数来模拟出 MA（q）序列，并分析其自相关图和偏自相关图的特征。

```
plot.arma<-function(data,main.text="",lag.max.num=9){
    plot(data, type="b",pch=16,main=paste(main.text," 序列散点图 "))
```

```
    acf(data,lag.max= lag.max.num,main=" 自相关图 ")
    pacf(data,lag.max= lag.max.num,main=" 偏自相关图 ")
}
par(mfcol=c(3,3))
data1<- arima.sim(n=500,list(ma=0.9))
plot.arma(data1, " MA(1)")
data2<- arima.sim(n=500, list(ma=c(0.7,-0.9)))
plot.arma(data2, " MA(2)")
data3<- arima.sim(n=500, list(ma=c(0.3,1,-0.9)))
plot.arma(data3, " MA(3)")
```

结果如图 5-21 所示。

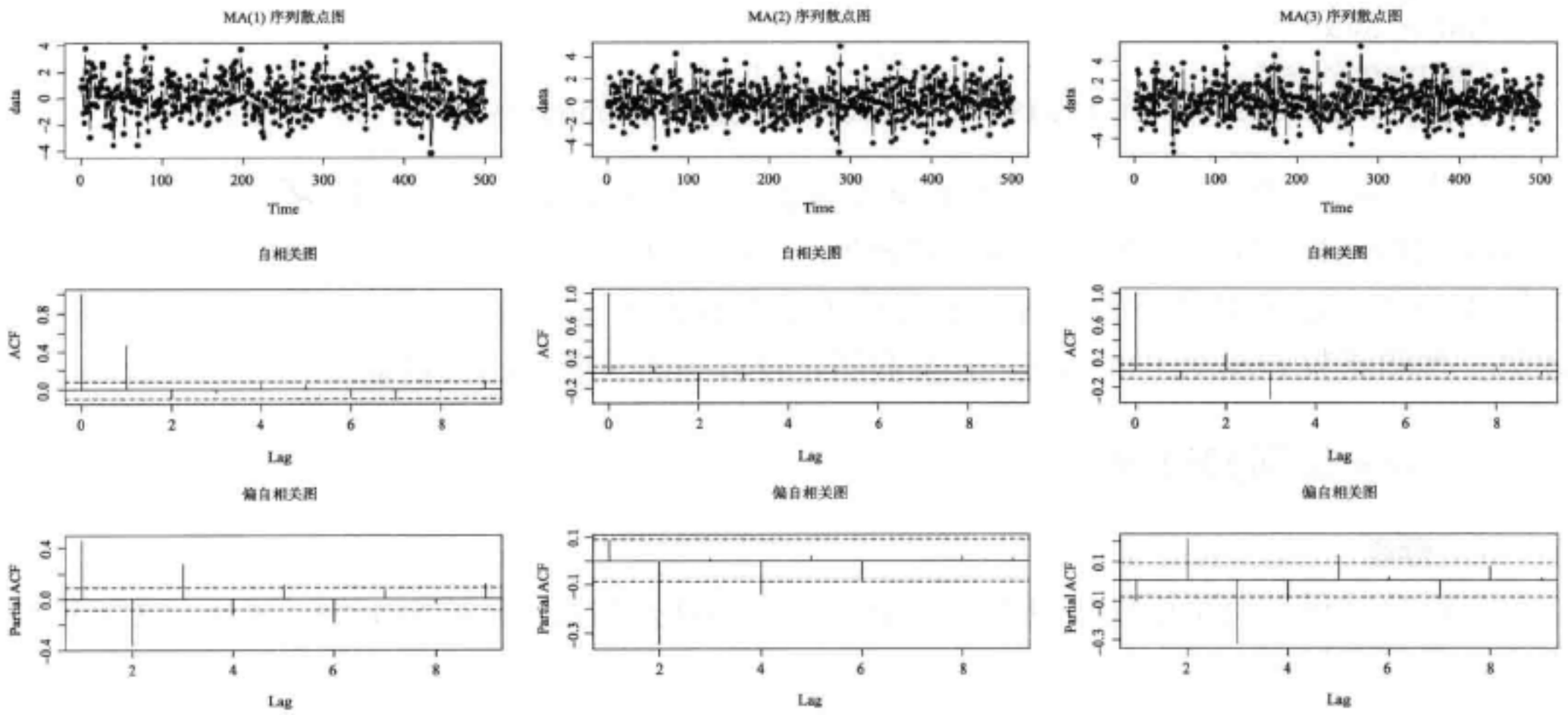
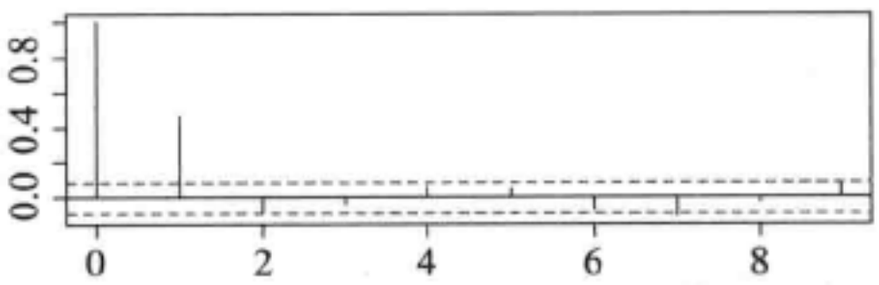
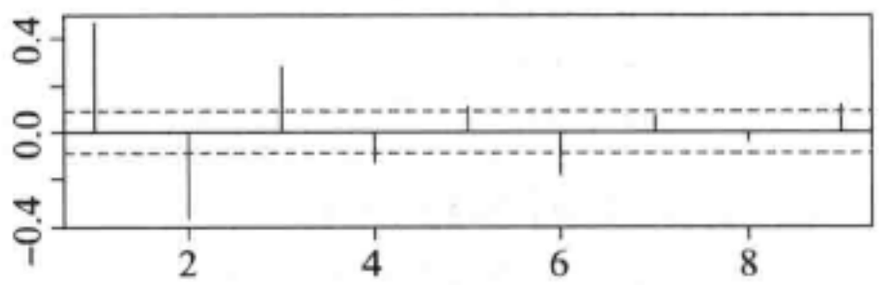


图 5-21　MA 序列的散点图、自相关图及偏自相关图

图 5-21 3 组图中的自相关图和偏自相关图的相同特性如表 5-5 所示。

表 5-5　MA 示例

	MA（q）的自相关图 acf	MA（q）的偏自相关图 pacf
相同特征	acf 呈指数级衰减，q 阶以后（不包括 q）的所有滞后序列，其相关性系数均处于 [−0.2,0.2] 范围内。我们称此类图形呈现“滞后 q 阶后截尾”形态	所有滞后序列的偏自相关系数逐步趋于 0，与阶数 q 无任何关系。我们称此类图形呈现“拖尾”形态
实例	MA（1）的自相关图  Lag 大于 1 时的滞后序列，其相关系数均处于蓝色虚线（[−0.2,0.2] 范围）以内	MA（1）的偏自相关图 

实际上，如果一个平稳序列的acf和pacf分别呈现出“截尾”和“拖尾”形态，则可以确定该序列为MA序列，并且可以使用acf的截尾滞后阶数来确定 q 的大小。

3. 参数 θ 的计算和预测

通过查看自相关图确定阶数 q 的取值后，就可以使用arima函数来计算参数 θ，并且可建立以MA过程为基础的模型。相关语句如下：

```
library(TSA)
sol.ma<- arima(data2, order = c(0, 0, 2))
predict(sol.ma,n.ahead=5)$pred
Time Series:
Start = 501
End = 505
Frequency = 1
[1]  0.297560910  -0.181454732  0.000367522  0.000367522  0.000367522
```

其中，arima中的order分别用于设置AR过程的阶数 p，差分过程的 d（用于稳定化）和MA过程的阶数 q。这里把 p 和 d 均设为0，表示只使用MA过程对序列建模。至于 θ 的实际大小并不是最为关心的，这里主要是为了得到模型sol.ma，进而使用predict函数进行预测。预测函数predict中的n.ahead参数用于设置预测新阶段的数据量。

5.5.4 AR自回归过程

1. 模型

自回归过程是使用序列本身作为变量，并对其进行加权（系数为1，ϕ_1，ϕ_2，…，ϕ_p）平均得出。其公式如下：

$$Y_t = e_t + \phi_1 e_{t-1} + \phi_2 e_{t-2} + \cdots + \phi_p e_{t-p} \tag{5-11}$$

上述公式为 p 阶自回归过程，记为AR(p)。

2. 相关图特征及阶数 p 的确定

下面仍使用arima.sim函数来模拟出AR（p）序列，并分析其自相关图和偏自相关图的特征。

```
par(mfcol=c(3,3))
data1<- arima.sim(n=500,list(ar=-0.4))
plot.arma(data1, " AR(1)")
data2<- arima.sim(n=500, list(ar=c(0.5,-0.25)))
plot.arma(data2, " AR(2)")
data3<- arima.sim(n=500, list(ar=c(0.4,0,-0.3)))
plot.arma(data3, " AR(3)")
```

最后结果如图5-22所示。

与MA过程的性质恰恰相反，如果一个平稳序列的acf和pacf分别呈现出“拖尾”和“截尾”形态，则可以确定该序列为AR序列，并且可以使用pcf的截尾滞后阶数来确定 p 的大小。

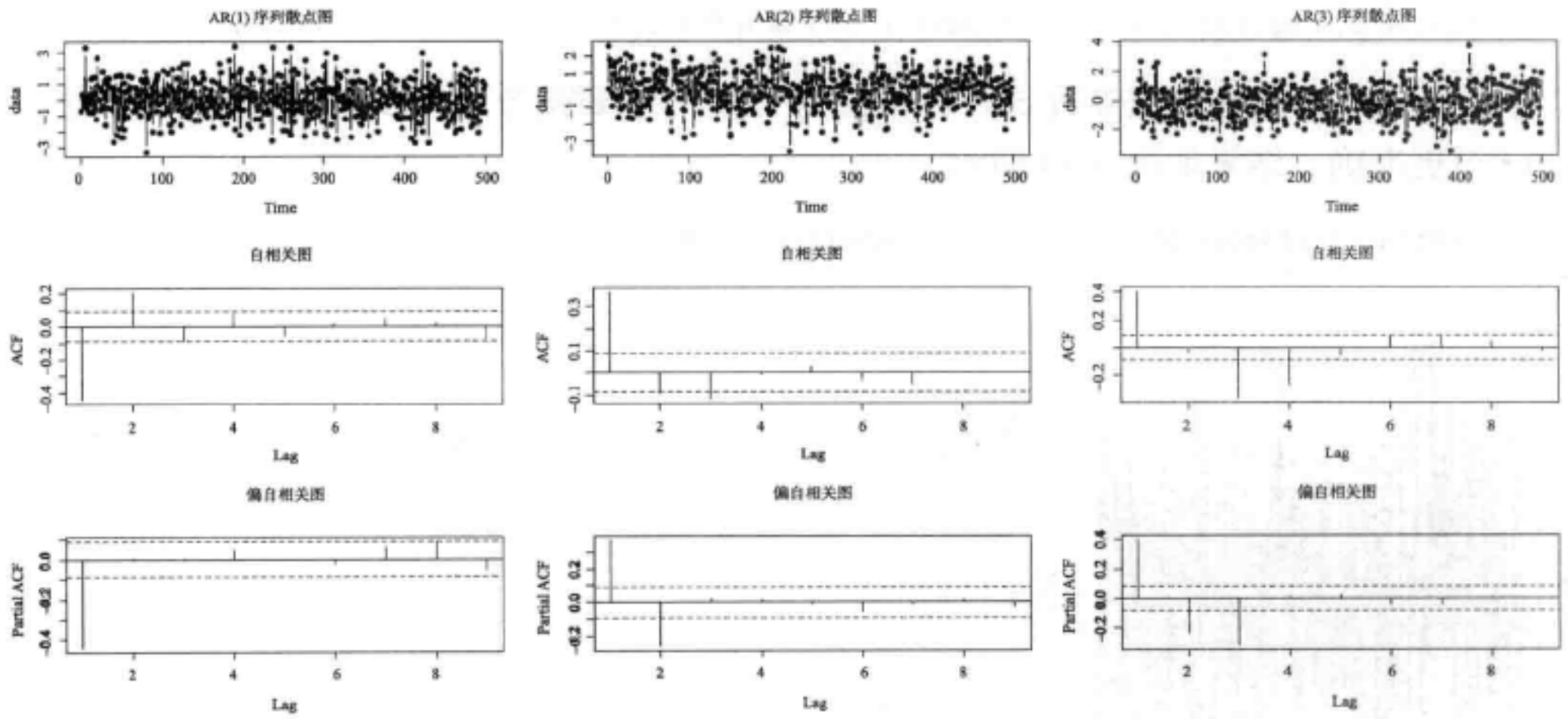


图 5-22　AR 序列的散点图、自相关图和偏自相关图

3. 参数 ϕ 的计算和预测

参考 5.5.3 节中的内容，确定 AR 过程的参数 ϕ。相关语句如下：

```
library(TSA)
sol.ar<- arima(data2, order = c(2, 0, 0))
predict(sol.ar,n.ahead=5)$pred
Time Series:
Start = 501
End = 505
Frequency = 1
[1]  0.17176372  -0.12987572  -0.06605051  0.04275739  0.02602923
```

实际上很少有指标数据可以单纯使用 AR 和 MA 过程来建立模型，更多的是使用 AR 和 MA 过程的综合建模 ARMA 或 ARIMA。

5.5.5　ARMA 自回归滑动平均混合过程

1. 模型

把滑动平均过程 MA 和自回归过程 AR 进行叠加就组成了自回归滑动平均混合过程 ARMA。其模型定义如下，并记为 ARMA (p,q)。

$$Y_t = e_t - \theta_1 e_{t-1} - \theta_2 e_{t-2} - \cdots - \theta_q e_{t-q} + \phi_1 e_{t-1} + \phi_2 e_{t-2} + \cdots + \phi_p e_{t-p} \tag{5-12}$$

2. 相关图的特征

下面模拟出 ARMA（1,1）的序列，并分析其自相关图和偏自相关图的特征。

```
par(mfcol=c(1,3))
data<- arima.sim(n=1000,list(ar=0.7,ma=-0.5))
plot(data, type="b",pch=16,main=" ARMA(1,1) 序列散点图 ")
acf(data,lag.max= 20, ci.type="ma",main=" ARMA(1,1) 自相关图 ")
```

```
pacf(data,lag.max=20,main=" ARMA(1,1) 偏自相关图 ")
```

注意，在 acf 函数中把参数 ci.type 设置为 "ma"，主要是考虑了 acf 的标准误的问题，以便 acf 图更准确。结果如图 5-23 所示。

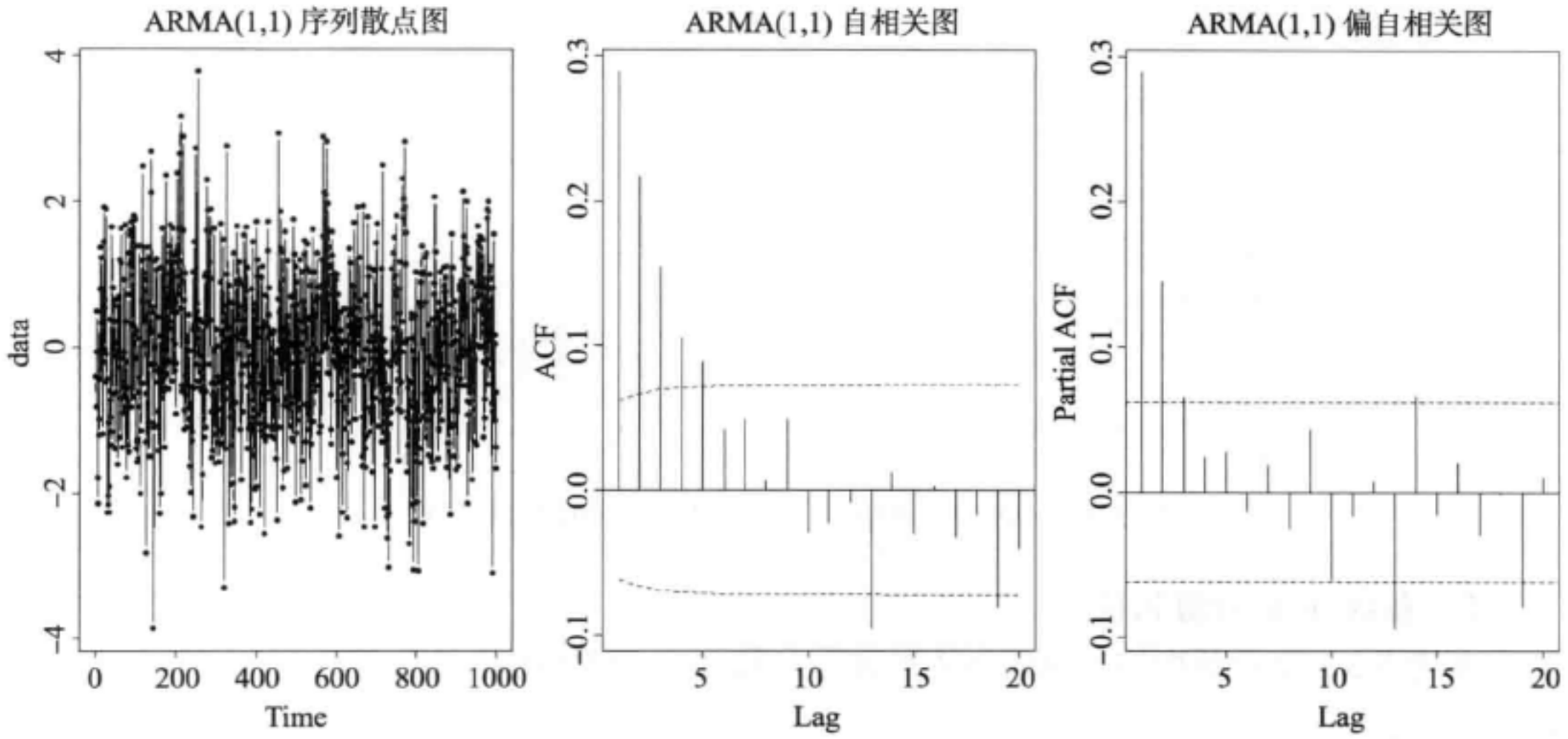
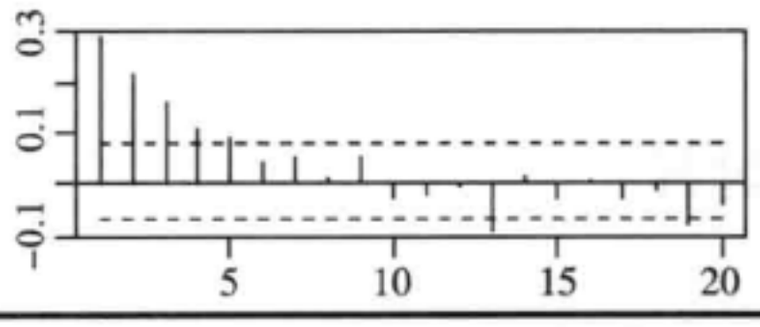
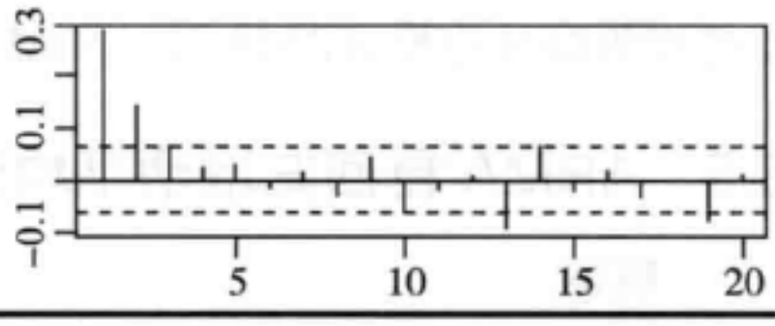


图 5-23 ARMA（1,1）的散点图、自相关图和偏自相关图

图 5-23 中的自相关图和偏自相关图的相同特性如表 5-6 所示。

表 5-6 ARMA 自相关图和偏自相关图的特征

	ARMA（p,q）的自相关图 acf	ARMA（p,q）的偏自相关图 pacf
相同特征	“拖尾”形态	
实例	ARMA（1, 2）的自相关图 	ARMA（1, 2）的偏自相关图 

在上例中，ARMA（1, 2）的自相关图以单侧（均大于 0）进行趋于 0 的拖尾。偏自相关图以双侧（大于和小于 0）进行趋于 0 的拖尾。对于 ARMA，已经不能单纯通过观察自相关图或偏自相关图来确定 p 和 q 了，因为此时它们已经重叠在一起。对于这种情况，可通过引入混合自相关图来解决。

3. 混合自相关图

混合自相关的原理这里不做过多描述，读者可以参考《时间序列分析及应用：R 语言》一书。这里只介绍在 R 语言中如何使用 eacf 函数来确定 ARMA 的 p 和 q。以上面代码中的 data2 为例，它实际是一个大致复合 ARMA（1,2）过程的序列。对于理想的 ARMA（1,2），

其 eacf 应该如图 5-24 所示。

把所有 *o* 聚在一起，并圈出一个三角形区域（黑色实线），其三角形顶点的行坐标名称为 *p* 的取值，列坐标名称是 *q* 的取值。图 5-24 中的三角形顶点在 2 行 3 列处，对应的行名称为 1，列名称为 2，即 *p*=1，*q*=2。接下来看看实际 data2 的 eacf 情况。

```
eacf(data2)
```

对应的结果如图 5-25 所示。

```
AR/MA
  0 1 2 3 4 5 6 7 8 9 10 11 12 13
0 x x x x x x x x x x x  x  x  x
1 x x o o o o o o o o o  o  o  o
2 x x x o o o o o o o o  o  o  o
3 x x x x o o o o o o o  o  o  o
4 x x x x x o o o o o o  o  o  o
5 x x x x x x o o o o o  o  o  o
6 x x x x x x x o o o o  o  o  o
7 x x x x x x x x o o o  o  o  o
```

图 5-24　理想 ARAM（1, 2）的 eacf

```
AR/MA
  0 1 2 3 4 5 6 7 8 9 10 11 12 13
0 x x o o o o o o o o o  o  o  o
1 x o o o o o o o o o o  o  o  o
2 x x o o o o o o o o o  o  o  o
3 x x x o o o o o o o o  o  o  o
4 x x o x o o o o o o o  o  o  o
5 x x x o x x o o o o o  o  o  o
6 x x x x x x x o o o o  o  o  o
7 x x x x x o x x o o o  o  o  o
```

图 5-25　data2 的 eacf 结果展示图

根据理想状态下 eacf 的情况来进行综合分析，初步判断有可能是如表 5-7 所示的两种情况。

表 5-7　混合自相关图示例

	情况 1	情况 2
三角形区域	AR/MA 0 1 2 3 4 5 6 7 8 9 10 11 12 13 0 x x o o o o o o o o o o o o 1 x o o o o o o o o o o o o 2 x x o o o o o o o o o o o 3 x x x o o o o o o o o o o 4 x x o x o o o o o o o o o 5 x x x o x x o o o o o o o 6 x x x x x x x o o o o o o 7 x x x x x o x x o o o o o	AR/MA 0 1 2 3 4 5 6 7 8 9 10 11 12 13 0 x x o o o o o o o o o o o 1 x o o o o o o o o o o o 2 x x o o o o o o o o o o 3 x x x o o o o o o o o o 4 x x o x o o o o o o o o 5 x x x o x x o o o o o o 6 x x x x x x x o o o o o 7 x x x x x o x x o o o o
p 和 *q*	*p*=1，*q*=2	*p*=1，*q*=1

具体选用哪组参数，需要分别比较建模后的质量才能确定（参见 5.5.6 节）。

```
sol.arma1<-arima(data2, order = c(1, 0, 2))
sol.arma2<-arima(data2, order = c(1, 0, 1))
```

5.5.6　检验模型质量

模型质量的衡量，主要是针对其残差值进行正态性和非自相关性的检验。残差满足正态性，主要是为了残差集中于某一数值，如果该值与 0 很接近，则它实际服从均值为 0 的正态分布，即说它是一个白噪声。残差满足非自相关性，主要是为了在残差中不再包括 AR 或 MA 过程产生的序列。

1. 正态性检验

在 4.1.2 节中已经介绍了正态性的检验，这里使用自定义的 norm.test 函数来对 5.5.5 节

中的模型进行检验。

```
source("D:/norm.test.r")
norm.test(sol.arma1$residuals)
          [1]" success: 服从正态分布, p.value= 0.221391691438416 > 0.05"
               Shapiro-Wilk normality test
          data:  input.data
          W = 0.9959, p-value = 0.2214
norm.test(sol.arma2$residuals)
          [1]" success: 服从正态分布, p.value= 0.212679901195716 > 0.05"
               Shapiro-Wilk normality test
          data:  input.data
          W = 0.9959, p-value = 0.2127
```

两个模型产生的残差均服从正态性，且 p=1，q=2 拟合模型的残差更具备正态性（p-value 较大）。

2. 非自相关性检验

在 5.5.1 节中介绍了如何使用 Box.test 函数进行残差的自相关性检验，相关语句如下：

```
diag.test<-function(sol.arma,p,q,max.lag.num){
    test.p<-0
    for(i in (p+q+1): max.lag.num){
        test.p[i]<-Box.test(sol.arma$residuals,i,fitdf=p+q,type="Ljung")$p.value
    }
    plot(c(rep(NA,p+q),test.p[-c(1:(p+q))]),main="残差的 Ljung 检验 p.value",ylim= c(0,1))
    abline(h=0.05,col="red")
    test.p
}
```

在自定义函数 diag.test 中，依次使用 Ljung 检验方式计算不同阶数滞后的 p-value。使用 Box.test 函数对残差进行检验时，要设置自由度参数 fitdf 为 p+q。代码如下：

```
diag.test(sol.arma1,1,2,20)
```

结果如图 5-26 所示。

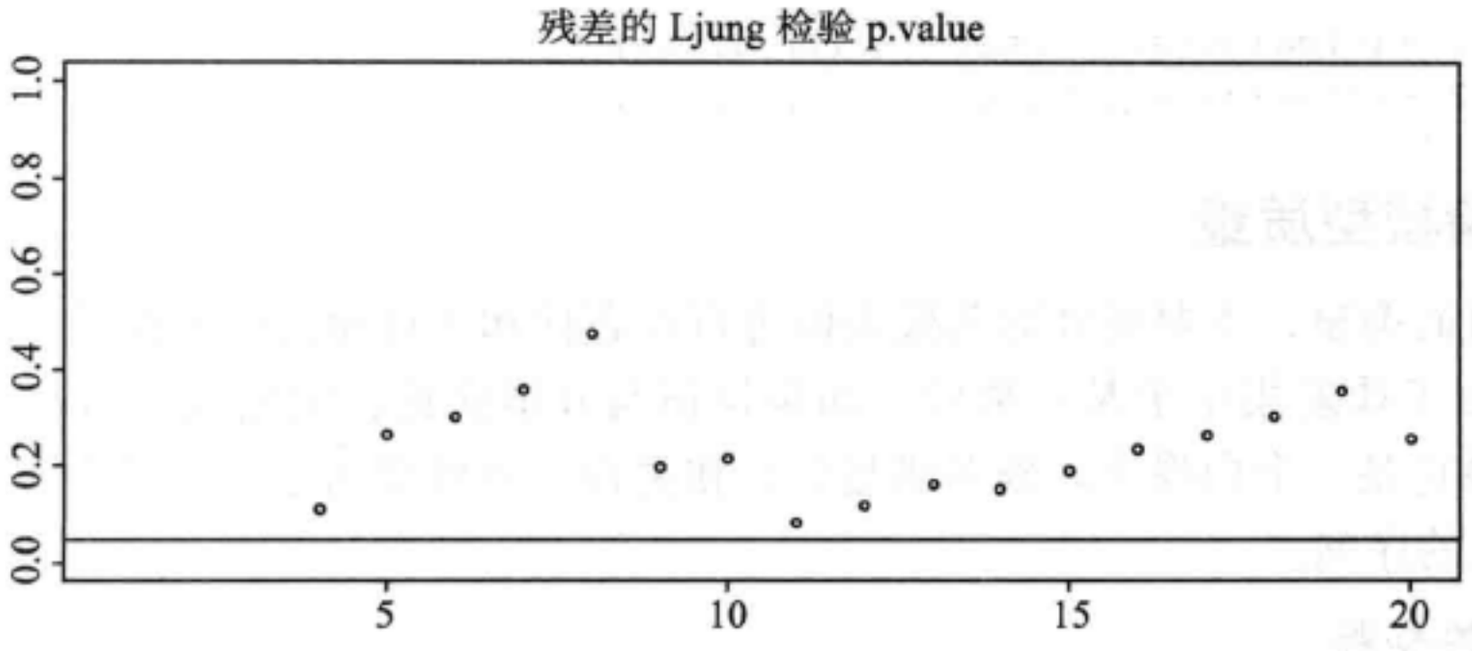


图 5-26 Ljung 检验结果展示图

由图 5-26 可见，20 阶以内的所有 Box.test 检验 p-value 值均大于 0.05，即均无自相关性。在 R 语言中，还可以使用 tsdiag 函数来快速完成上述绘图。代码如下：

```
tsdiag(sol.arma1)
```

图 5-27 中依次绘制了残差的散点图、自相关图和不同阶数下的 Box.test 检验 p-value 值。

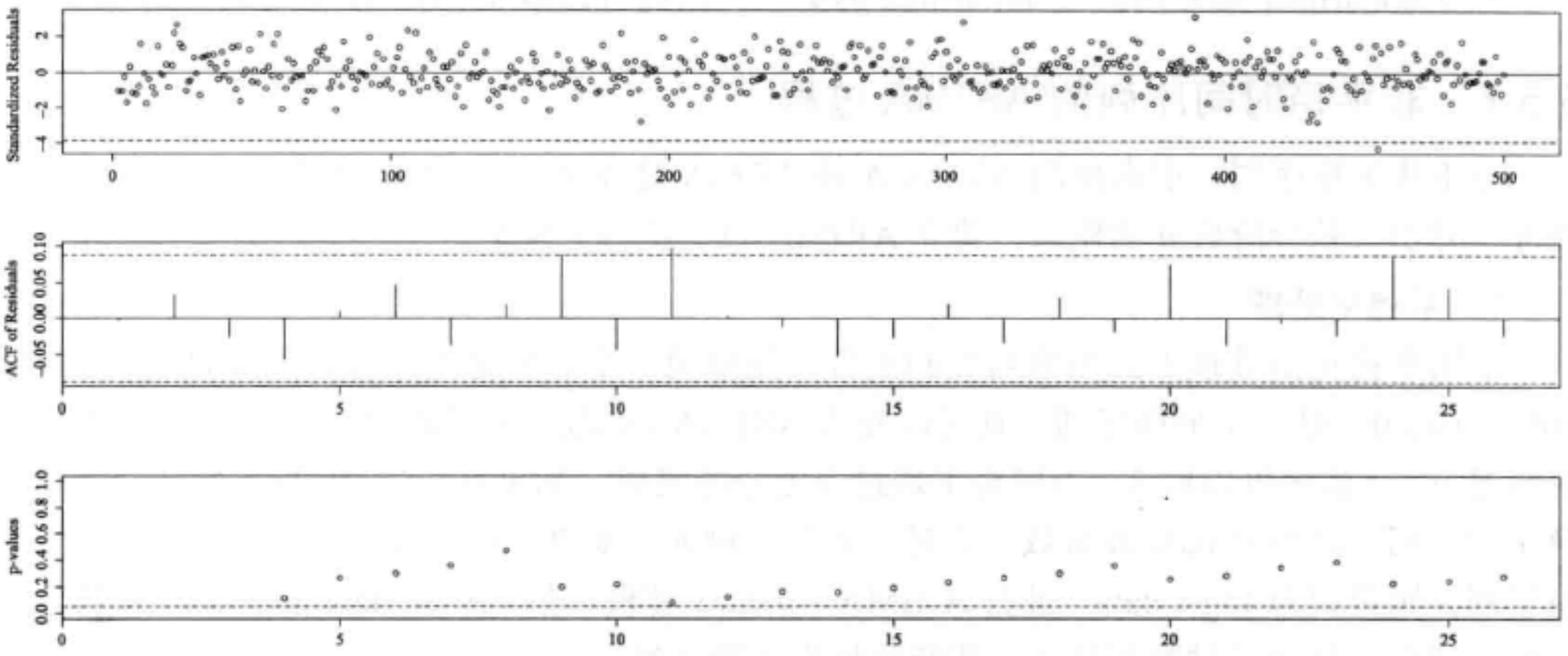


图 5-27　tsdiag 函数结果展示图 1

接着对 p=1，q=1 的模型 sol.arma2 进行非自相关性的检验。语句如下：

```
tsdiag(sol.arma2)
```

其结果如图 5-28 所示。

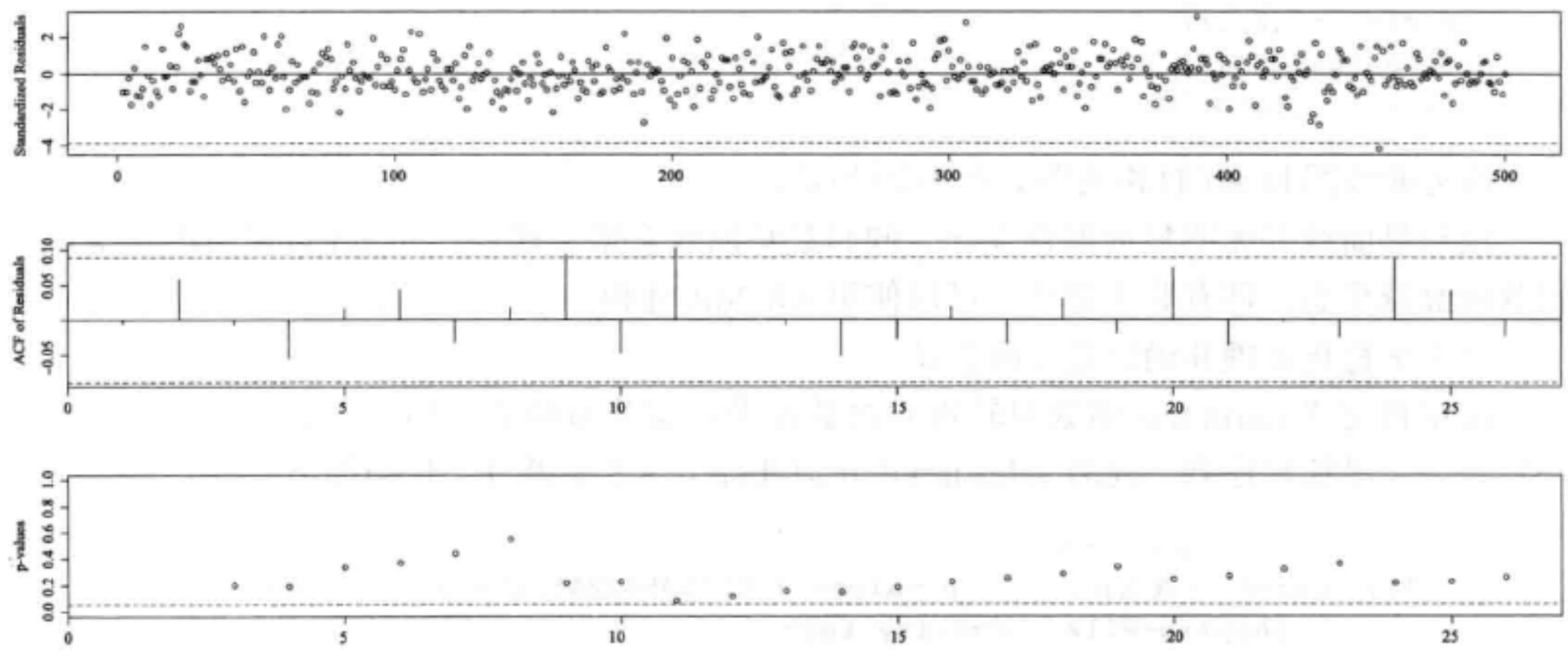


图 5-28　tsdiag 函数结果展示图 2

可见 sol.arma2 的残差也符合非自相关性，此时可以说 sol.arma1 模型和 sol.arma2 模型

均可以作为 data2 的 arma 模型。最后比较残差的均值。代码如下：

```
mean(sol.arma1$residuals)
    [1] -0.0002713136
mean(sol.arma2$residuals)
    [1] -0.002217462
```

可见 sol.arma1 残差均值比 sol.arma2 的更小，因此可以取 sol.arma1 作为最后的模型。

5.5.7 非平稳时间序列的 ARIMA 过程

对于非平稳序列，很难使用 AR、MA 或 ARMA 过程来建模。此时要先对其进行平稳化处理，并引入差分阶数 d 的概念，建立 ARIMA（p，d，q）模型。

1. 平稳化处理

常用平稳化的处理方式为取对数和差分。如果说一个非平稳指标数据 data 的 $d0$ 阶差分 diff（data,$d0$）是一个平稳序列，则可以使用 ARIMA（p,d,q）为其建模，其中 d 等于 $d0$。一般情况下，d 最多可以取 3，否则会出现过分差分的错误。如果 diff（data,d0）仍是一个非平稳序列，则可以先对 data 取对数，再进行差分。例如，如果 diff（log（data）, $d0$）是一个平稳序列，则可以对 log（data）进行 ARIMA（p,d,q）建模，然后得到 data 的模型。大部分的非平稳序列，经过取对数和差分后都能转化为平稳序列。

2. 实例：ARIMA 为销售指标建模

ARIMA 建模就是在 ARMA 的基础上确定差分阶数 d。这里继续以 5.1 节的销售数据为例，由于 ARIMA 不能处理周期性数据，所以先分解销售数据 sales.ts，并使用 ARIMA 对长期趋势 trend 进行建模。相关语句如下：

```
sales.trend<-stl(sales.ts,"per")$time.series[,"trend"]
par(mfcol=c(2,1))
plot(sales.trend)
acf(sales.trend,20)
```

趋势曲线图和 acf 自相关图如图 5-29 所示。

该趋势曲线并无明显周期性变化，同时趋势曲线有增长趋势，而 acf 自相关图未呈现出指数级衰减形态，即有非平稳性。可以使用 ARIMA 建模。

（1）平稳化处理并确定差分阶数 d

使用自定义 norm.test 函数对转换后的数据做正态分布检验，如果它服从正态分布，则一定是一个平稳性序列。先对 sales.trend 分别进行 1 ～ 3 阶差分。代码如下：

```
norm.test(diff(sales.trend))
    [1]" error: 不服从正态分布, p.value= 7.02750563291041e-05 <= 0.05"
            Shapiro-Wilk normality test
    data:  input.data
    W = 0.92, p-value = 7.028e-05
norm.test(diff(sales.trend,2))
```

```
    [1]" error: 不服从正态分布, p.value= 0.0002657970828482 66 <= 0.05"
            Shapiro-Wilk normality test
    data:  input.data
    W = 0.9308, p-value = 0.0002658
norm.test(diff(sales.trend,3))
    [1]" error: 不服从正态分布, p.value= 0.000350884161770 39 <= 0.05"
            Shapiro-Wilk normality test
    data:  input.data
    W = 0.9324, p-value = 0.0003509
```

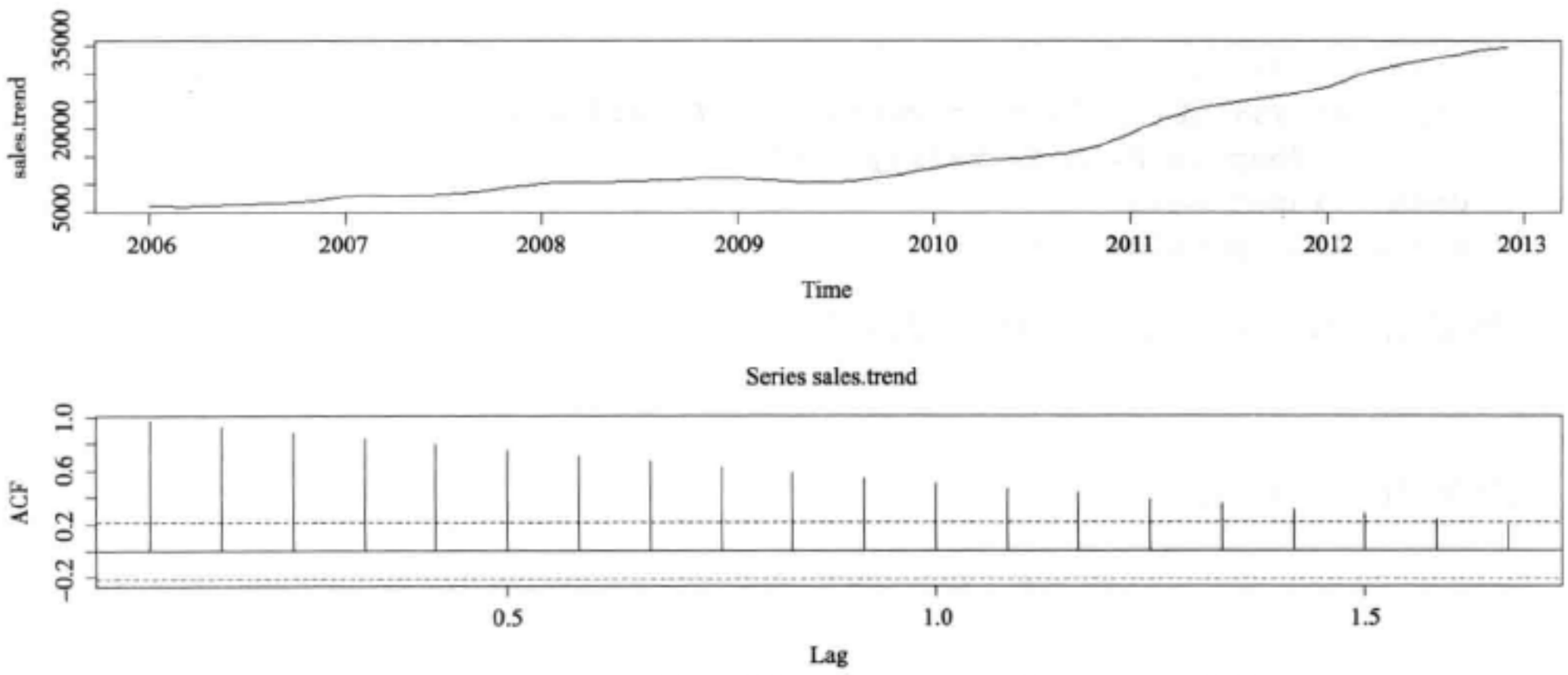


图 5-29　趋势曲线图和 acf 自相关图

由于其结果均不满足正态分布，所以继续尝试取对数转换。代码如下：

```
norm.test(diff(log(sales.trend)))
    [1]" success: 服从正态分布, p.value= 0.119218779015 38 > 0.05"
            Shapiro-Wilk normality test
    data:  input.data
    W = 0.9758, p-value = 0.1192
```

这里检验 p-value 值为 0.1192，大于 0.05，即表示服从正态分布，可确定出差分阶数 d 等于 1。然后建立临时变量 sales.trend.log 用于确认参数 p 和 q 的数值。相关语句如下：

```
sales.trend.log<-log(sales.trend)
```

（2）确定 p 和 q

绘制混合自相关图形 eacf 并观察可能的 p 和 q 数值。相关语句如下：

```
eacf(sales.trend.log)
```

结果如图 5-30 所示。

通过观察三角形顶点位置，可以暂时确定 p=1，q=1。

```
AR/MA
  0 1 2 3 4 5 6 7 8 9 10 11 12 13
0 x x x o o o x o o o o  o  o  x
1 x o o o o o x o o o o  o  o  x
2 o o o o o o o o o o o  o  o  o
3 o x o o o o o o o o o  o  o  o
4 o x o o x o o o o o o  o  o  o
5 o x o o x o o o o o o  o  o  o
6 x x x o x o o o o o o  o  o  o
7 x x o o x o o o o o o  o  o  o
```

图 5-30　结果展示

对于最终的模型，要在后续检验模型质量的过程中不断修正。

（3）使用 ARIMA 进行建模

确定 p=1，d=1，q=1 后，使用 ARIMA(p,d,q) 对 sales.trend.log 进行建模。相关语句如下：

```
sol<-arima(sales.trend.log,order=c(p=1,d=1,q=1))
```

（4）检验模型质量

参考 5.5.6 节，对模型和实际数据的残差数据进行正态性检验和非自相关性检验。代码如下：

```
norm.test(sol$residuals)
    [1]" success: 服从正态分布, p.value= 0.0740049062011937 > 0.05"
            Shapiro-Wilk normality test
    data:  input.data
    W = 0.973, p-value = 0.074
```

结果显示，模型 sol 的残差服从正态分布。

```
tsdiag(sol)
```

结果如图 5-31 所示。

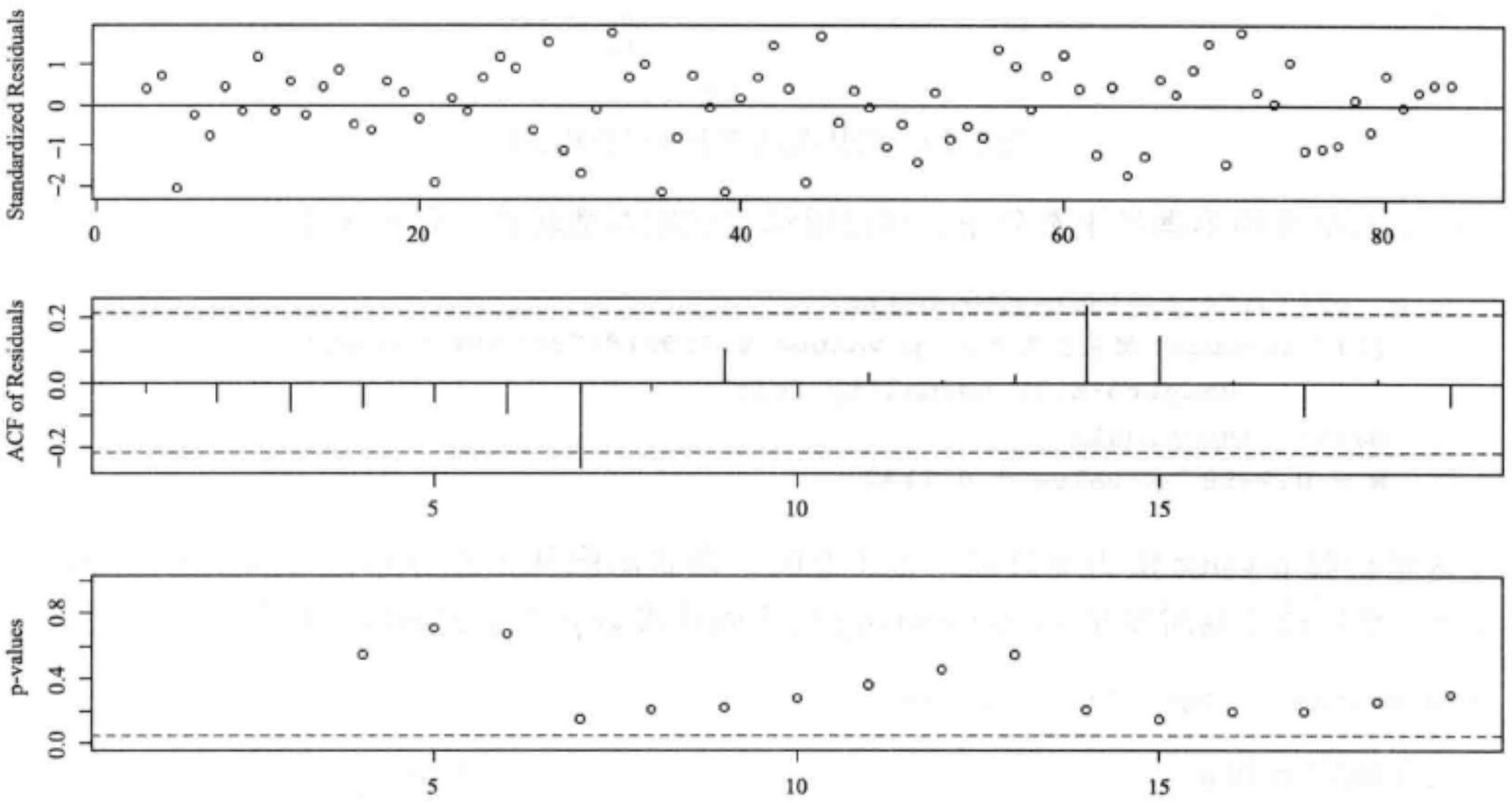


图 5-31 结果展示

由于 p-values 的数值图中所有点均处于红色曲线以上，即模型 sol 的残差数据无自相关性，所以 sol 可以作为 sales.trend.log 的 ARIMA 模型。

（5）预测

1）预测 sales.trend 趋势数据。

下面使用 predict 函数预测 2013 年 1 月至 12 月的 sales.trend.log 数据，使用 log 函数的反函数 exp 得到 sales.trend 的预测数据，并绘制散点图。代码如下：

```
pre<-exp(predict(sol,n.ahead=12)$pred)
    Time Series:
    Start = 85
    End = 89
    Frequency = 1
    [1] 35063.53  35265.56  35456.76  35424.34  35382.86  35441.89  35744.68
    [8] 35880.87  35985.20  36231.64  36490.87  36852.85
plot(c(sales.trend,pre),pch=16,type="b",col=c(rep("black",84),rep("red",12)))
```

结果如图 5-32 所示。

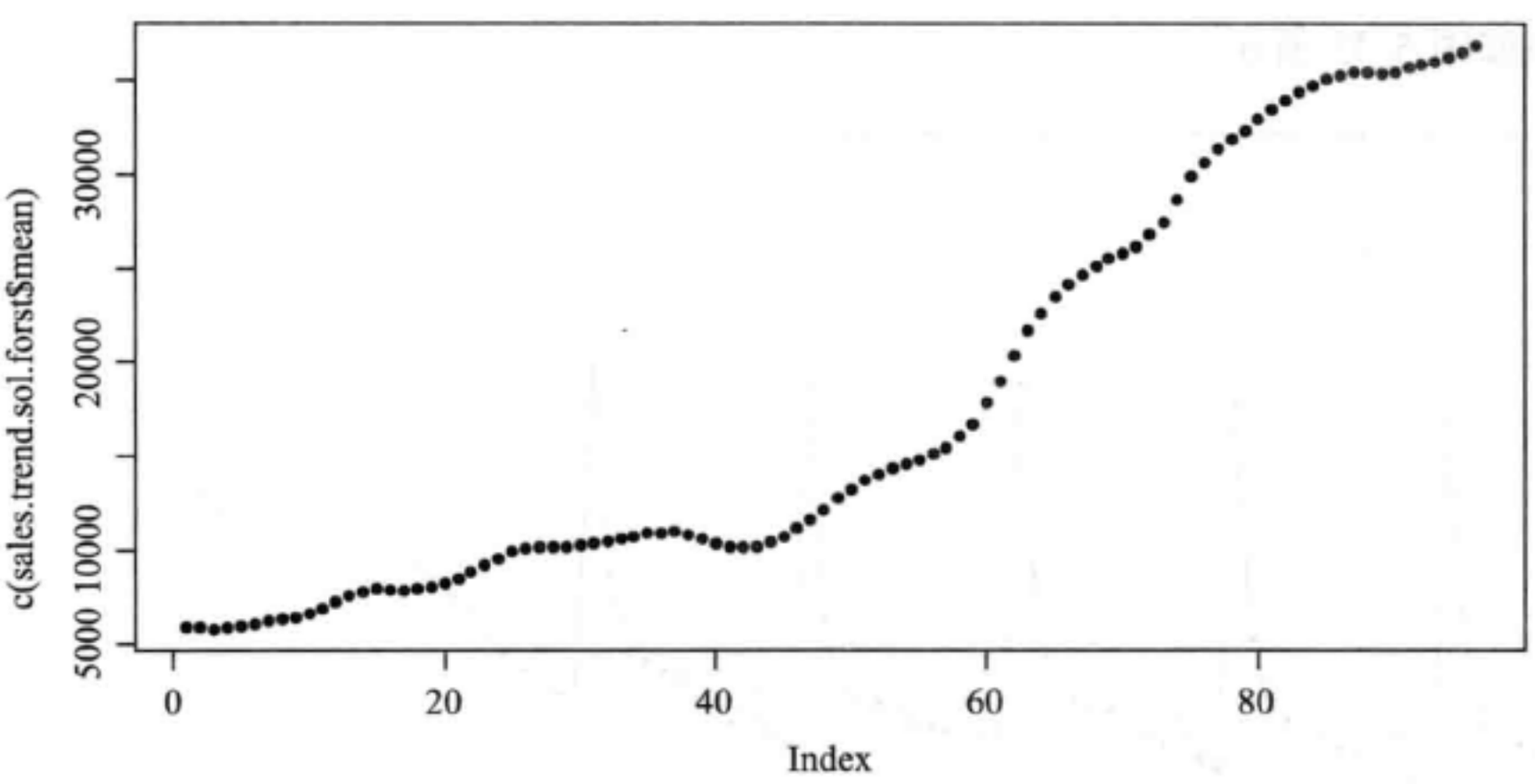


图 5-32　月销售数据的趋势预测图

其中黑色点为原始的 sales.trend 数据，红色点为最终预测的 2013 年全年数据。predict 函数还会给出标准误差 SE，以用于计算预测的范围。代码如下：

```
se<-exp(predict(sol,n.ahead=12)$se)
    Time Series:
    Start = 85
    End = 89
    Frequency = 1
    [1] 386.4628  776.2050  1246.7429  1785.4876  2384.5924  3038.4706
    [7] 3742.8634  4494.3788  5290.2305  6128.0757  7005.9084  7921.9853
```

最后根据公式：

预测范围 = 预测值 ± 置信度（alpha）* 标准误差 SE

计算不同置信度下的预测范围，如计算 95% 置信度下的预测范围 uper 和 lower。

```
uper<-pre+0.95*se;
lower<- pre-0.95*se;
```

2）预测 sales.ts 销售数据。

上面已经预测了趋势数据 sales.trend，在此基础上添加使用 stl 分解的周期性数据 sales.seasonal。代码如下：

```
sales.seasonal<-stl(sales.ts,"per")$time.series[,"seasonal"]
sales.pre<-pre+ sales.seasonal[1:12]                    # 预测值
sales.pre.uper<- uper+ sales.seasonal[1:12]             #95% 置信度下的范围上限
sales.pre.lower<- lower + sales.seasonal[1:12]          #95% 置信度下的范围下限
```

最后绘制出散点图。代码如下：

```
sales.seasonal<-stl(sales.ts,"per")$time.series[,"seasonal"]
plot(c(sales.trend,pre)+c(sales.seasonal ,sales.seasonal[1:12]),type="b",pch=16,
     col=c(rep("black",84),rep("red",12)))
```

结果如图 5-33 所示。

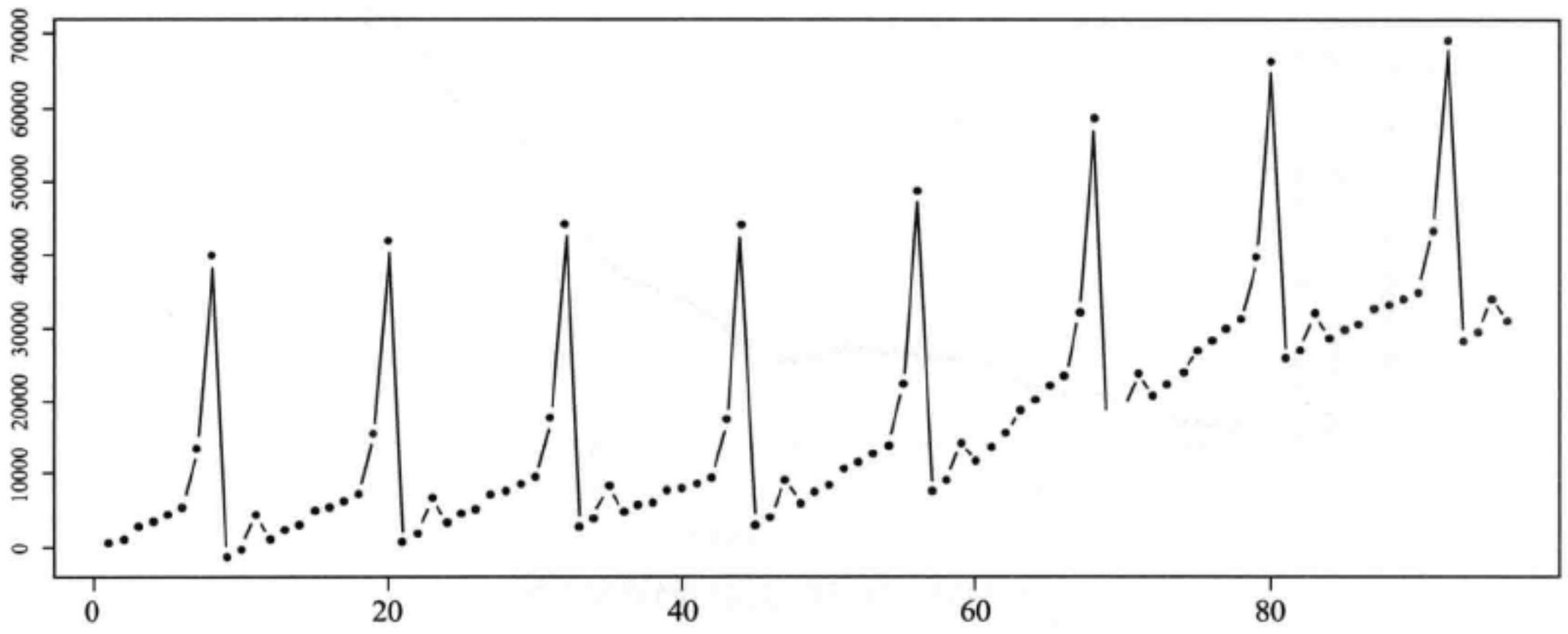


图 5-33 月销售数据的预测图

注意

常见的非周期性指标数据基本都可以通过 ARIMA 模型进行预测和建模，并且 p、q 和 d 的取值均不会过高（基本为 0、1、2、3）。

第 6 章

连续指标建模：回归分析

数据指标的建模指的是，使用若干自变量并建立公式，以预测目标变量（因变量）。

如果研究的目标变量是连续型的，则称其为回归分析（将在本章进行阐述）；如果是分类型，则称其为分类分析（将在下一章讲解）。表 6-1 是针对这两类分析给出的说明。

表 6-1　回归分析和分类分析说明

目标变量	=*f*（自变量）	名称	R 语言典型函数
连续型变量	=*f*（连续型变量，分类型变量…）	回归分析	lm、step 等
分类型变量	=*f*（连续型变量，分类型变量…）	分类分析	ctree 、J48、randomforest、bnlearn 等

6.1　一元线性回归分析

首先介绍回归分析中最基础的情况：一元线性回归分析。它规定模型 *f* 函数只能是 $y=k\times x+b$ 的形式，即只使用一个变量 x（故称为一元）的线性形式来预测目标变量 y。

6.1.1　引例

表 6-2 为某电商网站历次促销活动中促销让利费用和销售金额的数据（单位是十万元），将使用该数据集来说明线性回归分析的应用。

表 6-2　数据集

样本	促销让利费用	促销销量	样本	促销让利费用	促销销量	样本	促销让利费用	促销销量
1	1.8	104	6	2.5	134	11	1.5	101
2	1.2	68	7	1.5	87	12	0.7	46
3	0.4	39	8	1.2	77	13	1.0	52
4	0.5	43	9	1.6	102	14	0.8	33
5	2.5	127	10	1.0	65			

使用如下语句来绘制其散点图：

```
cost=c(1.8,1.2,0.4,0.5,2.5,2.5,1.5,1.2,1.6,1.0,1.5,0.7,1.0,0.8)
sales=c(104,68,39,43,127,134,87,77,102,65,101,46,52,33)
data<-data.frame(cost=cost,sales=sales)
plot(data,pch=16,xlab=" cost 促销让利费用（十万元）",ylab=" sales 促销销量（十万元）")
sol.lm<-lm(sales ~ cost,data)
abline(sol.lm,col="red")
```

绘制的散点图如图 6-1 所示。

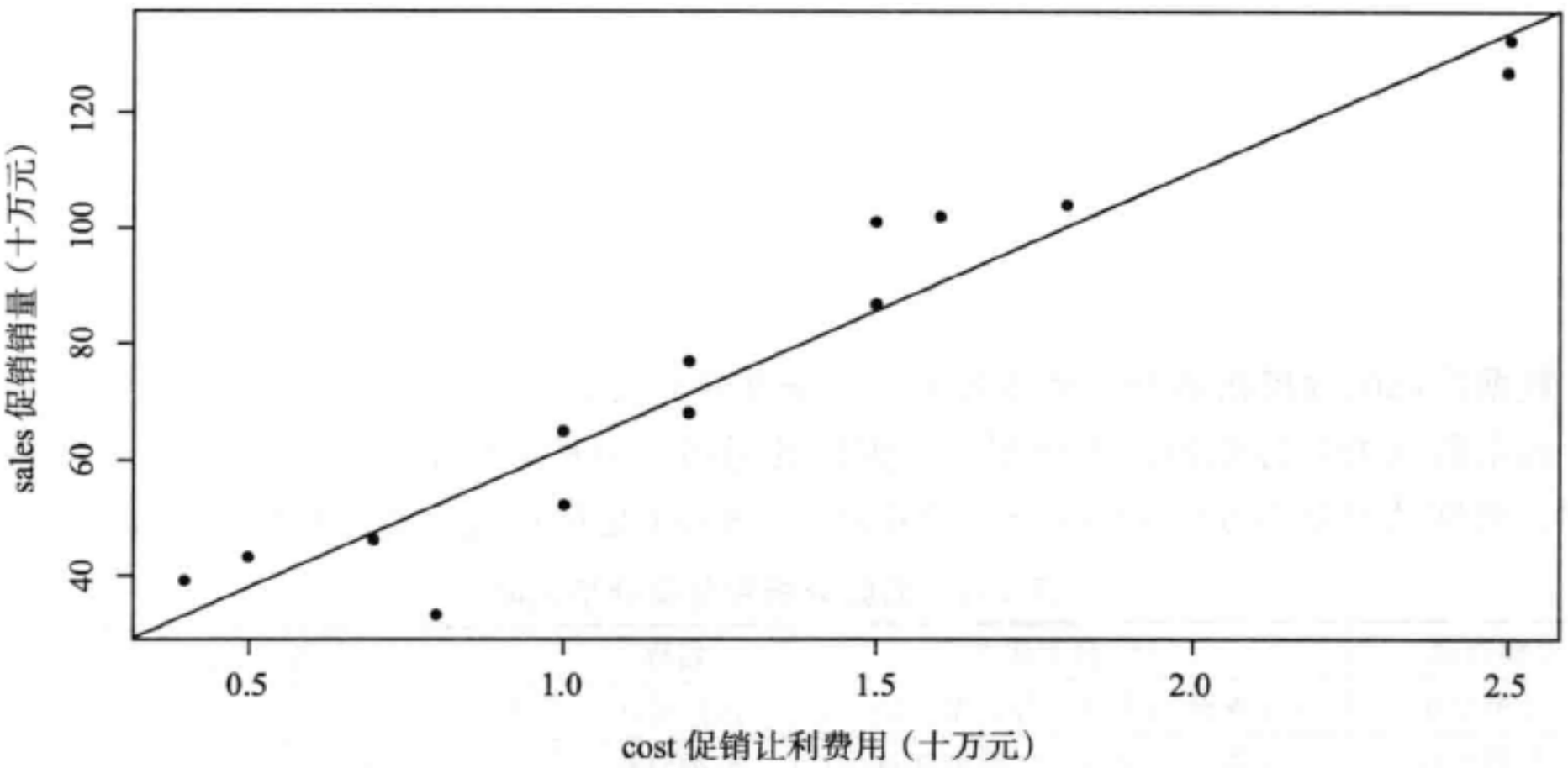


图 6-1 费用—销量回归散点图

由图 6-1 可观察到，促销让利费用 cost 变量和促销销量 sales 变量之间呈直线形式，这里绘制了一条红色直线（sales.sol = $k \times$ cost + b）横穿散点图。此时 sales=sales.sol+ε= $k \times$ cost+b+ε，其中 ε 是模型误差。可以预见，ε 的整体均值越小且越接近正态分布，则模型参数 k 和 b 越合理。

6.1.2 一元线性回归分析的原理及 R 语言实现

1. 最小二乘法

计算线性回归模型 sales.sol= $k \times$ cost + b 的基本思想是使用最小二乘法。

如果取一个样本点 i，它所对应的实际指标数据为 ($cost_i$，$sales_i$)，模型的预测数据为 ($cost_i$，$sales.sol_i$)，模型误差为 $sales.sol_i - sales_i = \varepsilon_i$。那么，在模型误差 ε_i 的平方和最小的情况下，计算模型参数 k 和 b，就称为最小二乘法（如图 6-2 所示）。最小二乘法旨在使获得的模型最大限度地拟合样本数据。

2. 样本方差和协方差

在第 5 章中介绍了样本方差 S_{xx}、S_{yy} 和协方差 S_{xy}，其定义分别如下：

变量 x 和 y 的样本均值分别为：

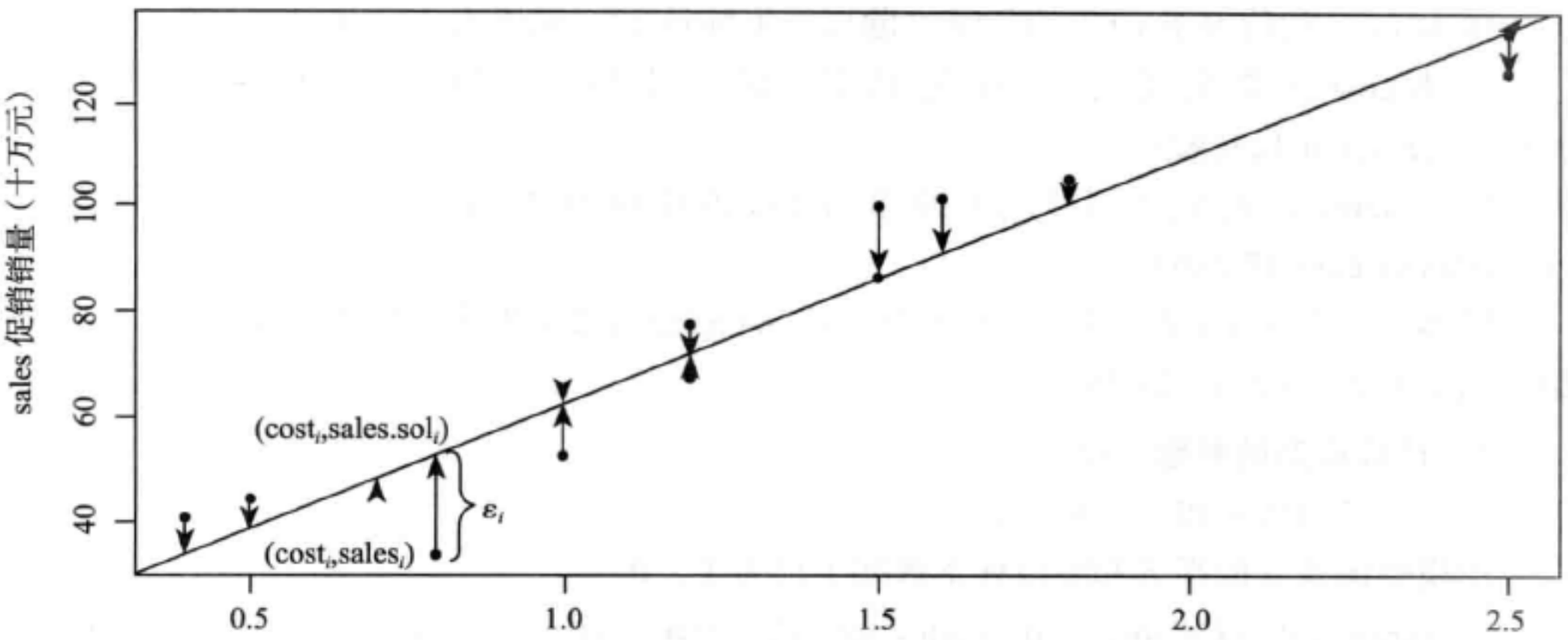


图 6-2 取样本点 i 的示例

$$\bar{x}=\frac{1}{n}\sum_{i=1}^{n}x_i,\quad \bar{y}=\frac{1}{n}\sum_{i=1}^{n}y_i$$

变量 x 的样本方差为：

$$s_{xx}=\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2$$

变量 y 的样本方差为：

$$s_{yy}=\frac{1}{n-1}\sum_{i=1}^{n}(y_i-\bar{y})^2$$

变量 x 和 y 的样本协方差为：

$$s_{xy}=\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})$$

上述公式会在后续章节中使用，由样本计算得到样本协方差的过程如表 6-3 所示。

表 6-3 计算协方差过程

样本	x(cost)	y(sales)	$x-\bar{x}$	$y-\bar{y}$	$(x-\bar{x})^2$	$(y-\bar{y})^2$	$(x-\bar{x})(y-\bar{y})$
1	1.8	104	0.5	27	0.25	729	13.5
2	1.2	68	−0.1	−9	0.01	81	0.9
3	0.4	39	−0.9	−38	0.81	1444	34.2
				……		.	
11	1.5	101	0.2	24	0.04	576	4.8
12	0.7	46	−0.6	−31	0.36	961	18.6
13	1.0	52	−0.3	−25	0.09	625	7.5
14	0.8	33	−0.5	−44	0.25	1936	22
总计	18.2	1078			5.56	14126	270.2

样本 cost 均值等于 x 的总计 18.2 除以样本数量 14，最终为 mean(cost)=1.3。

样本 sales 均值等于 y 的总计 1078 除以样本数量 14，最终为 mean(sales)=77。

样本 cost 方差 S_{xx} 等于 $(x-x)^2$ 的总计 5.56 除以样本数量减 1(14–1=13)，最终为 cov(cost,cost)=0.4276923。

样本 sales 方差 S_{yy} 等于 $(y-y)^2$ 的总计 14126 除以样本数量减 1(14–1=13)，最终为 cov(sales,sales)= 1086.615。

样本 cost 和 sales 协方差 S_{xy} 等于 $(x-x)(y-y)$ 的总计 270.2 除以样本数量减 1(14–1=13)，最终为 cov(cost,sales)=20.78462。

3. 计算模型的参数 k 和 b

（1）估算参数 k 和 b 的典型值

把模型误差 ε_i 的平方和除以样本数减 1 记为 S_e，即

$$S_e=\frac{(\text{sales.sol}_1-(k\times\text{cost}_1+b))^2+(\text{sales.sol}_2-(k\times\text{cost}_2+b))^2+\cdots+(\text{sales.sol}_{14}-(k\times\text{cost}_{14}+b))^2}{14-1}$$

$$=S_{yy}-k\times S_{xy}$$

如果 S_e 最小，则对 k 和 b 的微分为 0，即 $d(S_e)/d(k)=d(Se)/d(b)=0$。经过公式演算可以得到：

$S_{xx}\times k=S_{xy}$；$\bar{y}=k\times\bar{x}+b$，即：

$$k=S_{xy}/S_{xx}=\text{cov(cost,sales)/cov(cost,cost)}=20.78462/0.4276923=48.59713$$

$$b=\bar{y}-k\times\bar{x}=\text{mean(sales)}-48.59713*\text{mean(cost)}=77-48.59713\times1.3=13.82373$$

计算上述参数的代码如下：

```
k<-cov(cost,sales)/cov(cost,cost)# 模型参数 k
b<- mean(sales)- k*mean(cost) # 模型参数 b
```

也可使用回归方程建立函数 lm 来计算。代码如下：

```
sol.lm<-lm(formula=sales ~ cost,data);sol.lm
Call:
lm(formula = sales ~ cost, data = data)
Coefficients:
(Intercept)         cost
      13.82        48.60
```

上面代码中设置 lm 的 formula 参数为 sales~cost，即欲使用 lm 函数建立 $k\times$ cost + b 形式的模型。在得到的结果中，Coefficients 是模型参数序列，其中 Intercept 是 k（可使用 sol.lm$coefficients[2] 或 summary(sol.lm)$coefficients[2,1] 读取），另一列 cost 是 b（可使用 sol.lm$coefficients[1] 或 summary(sol.lm)$coefficients[1,1] 读取）。

使用 lines 函数可以在散点图中直接绘制出回归直线，最后得到 6.1.1 节中的图形（图 6-1）。代码如下：

```
plot(data,pch=16,xlab="cost 促销让利费用（十万元）",ylab="sales 促销销量（十万元）")
sol.lm<-lm(sales ~ cost,data)
```

```
abline(sol.lm,col="red")
```

（2）估算参数 k 和 b 的取值范围

在计算得到的回归模型 $y=k\times x+b$ 中，系数 b 和 k 只是一个真实模型的典型估算值，其估算范围是：

$[k_i-\text{sd}(k_i)t_{a/2}(n-2), k_i+\text{sd}(k_i)t_{a/2}(n-2)]$

其中，k_0 表示回归模型 $y=k\times x+b$ 中的 b，k_1 表示 k。$\text{sd}(k_i)$ 是参数的标准差，可以使用 summary(sol.lm)$coefficients[,2] 直接读取。代码如下：

```
df<- sol.lm$df.residual
alpha=0.05
left<- summary(sol.lm) $coefficients[,1]-summary(sol.lm) $coefficients[,2]*qt(1-
alpha/2,df);left
    (Intercept)          cost
      1.667702     40.182861
    right<- summary(sol.lm) $coefficients[,1]+summary(sol.lm) $coefficients[,2]*qt(1-
alpha/2,df);right
    (Intercept)          cost
      25.97978     57.01138
```

在上述代码中，*df* 为自由度，等于样本数 n 减去自变量数 p，再减 1，即为 $n-2$，该自由度可以通过 sol.lm$df.residual 直接读取。alpha 是置信度，典型取值是 0.05，left 是取值范围的最小值，right 是取值范围的最大值。最后得到的结果为：在置信度为 0.05 的情况下，参数 b 的范围为 [1.667702,25.97978]，k 的范围为 [40.182861，57.01138]。

注意

在多元 (p 元) 回归分析模型 $y=k_1\times x_1+k_2\times x_2+\cdots+k_p\times x_p+b$ 中，参数 k_i 和 b 的取值范围是

$$[k_i-\text{sd}(k_i)t_{a/2}(n-p-1), k_i+\text{sd}(k_i)t_{a/2}(n-p-1)]$$

其中，p 是自变量数，n 是样本数，计算在 alpha 置信度下参数范围的 R 语言代码如下：

```
df<- sol.lm$df.residual;alpha=0.05
left<- summary(sol.lm) $coefficients[,1]-summary(sol.lm) $coefficients[,2]*qt(1-
alpha/2,df)
right<- summary(sol.lm) $coefficients[,1]+summary(sol.lm)
$coefficients[,2]*qt(1-alpha/2,df)
```

4. 衡量相关程度

上面介绍了线性回归分析的原理和公式，下面介绍如何使用相关系数 r 来衡量目标变量与自变量之间的相关程度。

（1）相关系数 r 和判定系数 r^2

变量 x 和 y 的样本相关系数为 $r_{xy}=\dfrac{s_{xy}}{\sqrt{s_{xx}}\sqrt{s_{yy}}}$，它用于衡量变量 x 和变量 y 间的相关性，其取值范围为 [-1，1]。如果 x 变量恒等于 y（完全正相关），则 $r_{xy}=1$；如果 x 恒等于 $-y$（完全

负相关），则 $r_{xy}=-1$；如果 x 和 y 的关系很小，则 $S_{xy}=0$，即 $r_{xy}=0$。可见 r_{xy}^2 趋近于 1 时，x 和 y 有较强的相关性（正相关 / 负相关性），而 r_{xy}^2 趋近于 0 时，则无相关性，因此称 r_{xy}^2 为判定系数：

$$r_{xy}^2=S_{xy}^2/(S_{xx}*S_{yy})=k*\ S_{xy}\ /\ S_{yy}=1-\ S_e/S_{yy}$$

对于引例（促销费用和销售金额的关系），可以使用如下代码来计算相关参数。

```
r<-cor(cost,sales)# 相关系数 r
r2<- r^2 # 判定系数 r^2
```

在 lm 函数产生的结果中也包含了判定系数 r 的信息。

```
summary(sol.lm)
Call:
lm(formula = sales ~ cost, data = data)
 <中间省略>
Residual standard error: 9.106 on 12 degrees of freedom
Multiple R-squared:  0.9296,    Adjusted R-squared:  0.9237
F-statistic: 158.4 on 1 and 12 DF,  p-value: 2.843e-08
```

上述结果中的 Multiple R-squared 就是判定系数 r^2，该系数也可以使用 summary(sol.lm)$r.squared 来直接读取。

（2）修正判定系数 adjusted.r^2

事实上，判定系数 r^2 在用于多元（多个自变量）回归分析时有一个缺点，自变量数越多，判定系数 r^2 越大。为了消除这个缺点，可以引入修正判定系数 adjusted.r^2 的概念，如表 6-4 所示。

表 6-4 判定系数说明

	判定系数 r^2	修正判定系数 adjusted.r^2
定义式	$r^2=1-\frac{S_e}{S_{yy}}$	$\text{adjusted.r}^2=1-\frac{\frac{S_e}{\text{样本个数}-\text{自变量个数}-1}}{\frac{S_{yy}}{\text{样本个数}-1}}$
特性	与自变量数成正比增长	公式中引入自变量数，消除其对判定系数的影响

lm 函数产生的结果中也包含了判定系数 r 的信息。

```
summary(sol.lm)
Call:
lm(formula = sales ~ cost, data = data)
 <中间省略>
Residual standard error: 9.106 on 12 degrees of freedom
Multiple R-squared:  0.9296,    Adjusted R-squared:  0.9237
F-statistic: 158.4 on 1 and 12 DF,  p-value: 2.843e-08
```

上述信息中的 Adjusted R-squared 就是修正判定系数，该系数也可以使用 summary(sol.lm)$adj.r.squared 来直接读取。

相关系数 r 或（修正）判定系数 r^2 均可衡量变量间的相关程度，相关系数的说明如表 6-5 所示。

表 6-5 相关系数说明

相关系数 r	$\|r\|=0$	$\|r\|<0.3$	$0.3<\|r\|<0.5$	$0.5<\|r\|<0.8$	$0.8<\|r\|$	$\|r\|=1$
（修正）判定系数 r^2	$r^2=0$	$r^2<0.09$	$0.09<r^2<0.25$	$0.25<r^2<0.64$	$0.64<r^2$	$r^2=1$
含义	完全不相关	不相关	低度相关	中度相关	高度相关	完全相关

5. 回归系数的显著性检验

（1）T 检验

前面提到，在所有计算得到的回归模型 $y=k\times x+b$ 中，系数 b 和 k 只是一个真实模型的估算值，其估算范围是 $[k_i-\mathrm{sd}(k_i)t_{a/2}(n-2),\ k_i+\mathrm{sd}(k_i)t_{a/2}(n-2)]$。这里要求模型参数 k 不可为 0，一旦为 0，该模型就无任何意义。T 检验可以检验各个模型参数是否等于 0，并计算其等于 0 时的概率。具体的原理这里不详细介绍，读者只要知道对每个模型参数使用 T 检验后计算得到 p.value 值越小，其值等于 0 的概率也就越小。一般来说，当 p.value 小于 0.05 时，可以认定 k 不会等于 0，即模型结果可用并通过了检验。

在 lm 函数产生的结果中也包含 T 检验信息：

```
summary(sol.lm)
Call:
lm(formula = sales ~ cost, data = data)
<中间省略>
Coefficients:
             Estimate    Std.Error    t value    Pr(>|t|)
(Intercept)  13.824      5.579        2.478      0.0291 *
cost         48.597      3.862        12.584     2.84e-08 ***
---
Signif. codes:  0‘***’ 0.001‘**’ 0.01‘*’ 0.05‘.’ 0.1‘ ’ 1
<后面省略>
```

上述信息中的 Pr(>|t|) 部分就是参数 b 和 k 的 T 检验 p.value 值。其中 b 的 p.value 值为 0.0291，k 的 p.value 值为 2.84e-08，由于其值均小于 0.05，因此参数 k 和 b 均通过了 T 检验。结果中的 Signif.codes 部分含义如表 6-6 所示。

表 6-6 p 值说明

0<p.value<0.001	0.001<p.value<0.01	0.01<p.value<0.05	0.05<p.value<0.1	0.1<p.value<1
展示 " *** "	展示 " ** "	展示 " * "	展示 " . "	展示 " "
显著性极强	显著性很强	显著性较强	弱显著性	显著性不通过
一般认为通过显著性检验			一般认为未通过显著性检验	

也可以使用如下代码直接读取结果：

```
summary(sol.lm)$coefficients[,4]
 (Intercept)          cost
2.907896e-02      2.843305e-08
```

其中 summary(sol.lm)$coefficients[1,4] 是参数 b 的 p.value 值，summary(sol.lm) $coefficients[2,4] 是参数 k 的 p.value 值。

另外，还可以通过观察参数 b 和 k 的范围来简单确定其是否可能等于 0，前面介绍过计算得到的参数 b 的范围是 [1.667702,25.97978]，k 的范围是 [40.182861，57.01138]，均不等于 0，即它一定会通过 T 检验，这与通过上面的代码得到的结论相符。

（2）F 检验

与 T 检验不同，F 检验用于在整体上检验模型参数是否为 0，并计算等于 0 的概率。其原理这里不详细介绍。F 检验的结果在 lm 函数产生的结果也有所体现。

```
summary(sol.lm)
Call:
lm(formula = sales ~ cost, data = data)
<中间省略>
Residual standard error: 9.106 on 12 degrees of freedom
Multiple R-squared:  0.9296,    Adjusted R-squared:  0.9237
F-statistic: 158.4 on 1 and 12 DF,  p-value: 2.843e-08
```

上述信息中，F-statistic 这一行的 p-value 部分就是参数 b 和 k 整体 F 检验的 p.value 值，其值为 2.843e-08，远小于 0.05，表示通过了 F 检验。在 summary(sol.lm) 中只给出了样本自由度、自变量自由度以及 F 值，可以使用如下代码来直接读取 F 检验的 p-value 值。

```
f<- summary(sol.lm)$fstatistic[1]
df1<- summary(sol.lm)$fstatistic[2]
df2<- summary(sol.lm)$fstatistic[3]
pf(f,df1,df2,lower.tail=F)
       value
2.843305e-08
```

或者

```
1- pf(f,df1,df2)
       value
2.843305e-08
```

6. 模型误差（残差）

回归模型 sol.lm 的误差（残差 residuals），可用于体现样本点模型预测值与实际数据的差异程度，可通过 sol.lm$residuals 直接读取误差数据。对于一个正确的回归模型，其误差要服从正态分布性，这里使用第 5 章中定义的 norm.test 函数：

```
norm.test(sol.lm$residual)
[1]" success: 服从正态分布, p.value= 0.832901236186661 > 0.05"
        Shapiro-Wilk normality test
data:  input.data
W = 0.9669, p-value = 0.8329
```

残差的标准误（Residual standard error）可以从整体上体现一个模型的误差情况，它可

以用于不同模型间性能的对比。在由 lm 函数获取的结果中，Residual standard error 一行就是相关数据：

```
summary(sol.lm)
Call:
lm(formula = sales ~ cost, data = data)
<中间省略>
Residual standard error: 9.106 on 12 degrees of freedom
Multiple R-squared:  0.9296,    Adjusted R-squared:  0.9237
F-statistic: 158.4 on 1 and 12 DF,  p-value: 2.843e-08
```

也可以使用 summary(sol.lm)$sigma 直接读取。

7. 预测

假设：促销让利费用为 $cost_i$，其销量 sales 服从均值为 $k \times cost_i + b$，标准差为残差标准误（residual standard error）的正态分布。在得到回归模型 sol.lm 后，可以给定一个新样本的自变量取值以及置信度，从而得到相应目标变量的取值范围。在 R 语言中，使用 predict 函数可以方便地计算预测值和取值范围。下面的代码返回原有 14 个样本的预测值数据。

```
predict(sol.lm)
       1        2        3        4        5        6        7
101.29856  72.14029  33.26259  38.12230 135.31655 135.31655  86.71942
       8        9       10       11       12       13       14
 72.14029  91.57914  62.42086  86.71942  47.84173  62.42086  52.70144
```

如果设定新样本点的自变量取值是 0.98 和 0.88，则通过如下代码可以得到预测值及取值范围。

```
new.data<-data.frame(cost=c(0.98,0.88))
sol.pre<-predict(sol.lm,new.data,level=0.95, interval="prediction");sol.pre
       fit      lwr      upr
1 61.44892 40.73626 82.16158
2 56.58921 35.75046 77.42796
```

对于新样本数据，必须使用数据框形式承载，并标明列名称。level 用于设置置信度取值，典型值是 0.95。参数 interval 设置为 "prediction" 表示结果要计算出取值区间。上例中的结果表明当 cost 取 0.98 时，预测值是 61.44892，预测范围是 [40.73626,82.16158]。当 cost 取 0.88 时，预测值是 56.58921，预测范围是 [35.75046,77.42796]。其中，置信度越高（预测精度越高），预测值的取值范围越大（预测值的可使用性越差）。

6.2 多元回归分析

在进行多元回归分析时，其中涉及的很多概念都可以直接承接一元回归分析中的说明。

6.2.1 引例

表 6-7 是某电商网站用户对 330 个商品的问卷调查数据，其中涉及了 7 个问题，每个问题的选项分 5 个等级。

表 6-7 问卷的问题描述

问题变量	问题描述	问题变量	问题描述
X1	价格比其他网站 / 店铺贵	X5	细节图片太少，对商品没有直观感受
X2	购买的人很少，感觉可能不太畅销	X6	商品评价一般
X3	感觉该商品质量和款式不值该价格，性价比低	Y	商品综合满意度
X4	商品款式不符合我的要求		

对每个商品的每个问题，累加 25 个用户的选择得分作为相关问题的最终得分，每个问题变量的取值范围为 [1,5] × 25=[25,125]，具体数据如表 6-8 所示。

表 6-8 问卷调查数据示例

商品	*X*1	*X*2	*X*3	*X*4	*X*5	*X*6	*Y*
商品 1~220 作为训练集数据							
1	51	30	39	61	92	45	43
2	64	51	54	63	73	47	63
…	…	…	…	…	…	…	…
199	70	46	57	75	85	46	65
220	58	68	54	64	78	52	50
商品 221~330 作为测试集数据							
221	40	33	34	43	64	33	50
…	…	…	…	…	…	…	…
330	82	39	59	64	78	39	82

接下来参考 6.1.2 节中的一元线性回归分析方法，并使用训练集和测试集数据对变量 *Y* 进行建模。

6.2.2 多元线性回归分析建模

与一元回归分析类似，只要重新设置 lm 函数的 formula 参数就可以实现如下多元回归建模代码：

```
sol.lm<-lm(formula=y~.,data.train)
```

这里把变量 *y* 作为目标变量，其余变量（使用点“.”来表示）作为自变量。具体实现方法如表 6-9 所示。

表 6-9 使用 summary(sol.lm) 代码得到的建模结果输出以及解释

1	R 输出	**Residuals:** **Min 1Q Median 3Q Max** **-12.6695 -4.5173 0.0195 4.5113 13.8389**
	解释	残差的 5 等分数值，可以使用 sol.lm$residual 代码直接读取模型残差

（续）

2	R 输出	**Coefficients:** **Estimate Std. Error t value Pr(>\|t\|)** **(Intercept) 13.81436 4.48931 3.077 0.00236 **** **x1 0.57413 0.05808 9.886 < 2e-16 ***** **x2 -0.04001 0.04273 -0.936 0.35024** **x3 0.32707 0.05497 5.950 1.09e-08 ***** **x4 -0.03845 0.07939 -0.484 0.62864** **x5 0.10373 0.04966 2.089 0.03790 *** **x6 -0.19553 0.06333 -3.088 0.00229 **** **---** **Signif. codes: 0‘ ***’ 0.001‘ **’ 0.01‘ *’ 0.05‘ .’ 0.1‘ ’ 1**
	解释	**Coefficients** 是模型参数的数值和显著性检验数据。 **Estimate** 列依次是模型 $y=k1\times x1+k2\times x2+\cdots+k6\times x6+b$ 中的 b，$k1$，k2，…，$k6$ 等参数的数据，并可使用 sol.lm$coefficients 或 summary(sol.lm)$coefficients[,1] 来直接读取。 **Pr(>\|t\|)** 列是 T 检验下各模型参数的 p.value 数据，并可使用 summary(sol.lm)$coefficients[,4] 来直接读取。通过结果可见 $k2$ 和 $k4$ 并未通过 T 检验
3	R 输出	**Residual standard error: 6.284 on 213 degrees of freedom**
	解释	残差的标准误为 6.284，可以使用 summary(sol.lm)$sigma 代码来直接读取
4	R 输出	**Multiple R-squared: 0.6422, Adjusted R-squared: 0.6321**
	解释	判别系数为 0.6422，可使用 summary(sol.lm)$r.squared 代码直接读取 修正后的判别系数为 0.6321，可使用 summary(sol.lm)$adj.r.squared 代码直接读取
5	R 输出	**F-statistic: 63.72 on 6 and 213 DF, p-value: < 2.2e-16**
	解释	F 检验的 p.value 值远远小于 0.05，即通过了 F 检验。也可以直接使用如下代码获取 F 检验的 p.value： **f<- summary(sol.lm)$fstatistic[1]** **df1<- summary(sol.lm)$fstatistic[2]** **df2<- summary(sol.lm)$fstatistic[3]** **pf(f,df1,df2,lower.tail=F)** 或 **1-pf(f,df1,df2)**

6.2.3 模型修正函数 update()

update 函数可以在 lm 模型结果的基础上任意添加或减少自变量，或对目标变量做取对数及开方等建模，其基本形式为 update(object, formula)，参数说明如表 6-10 所示。

表 6-10 updata 函数参数说明

参数	说明		
Object	由 lm 函数得到的模型结果，如本节引例中的 sol.lm		
Formula	设置新的修正模型，使用点“.”代表旧模型 sol.lm 中 formula 的相应部分。如：		
	1	修正模型	在模型中增加 $x2$ 平方变量： $y=k_1*x_1+k_2*x_2+k_3*x_3+k_4*x_4+k_5*x_5+k_6*x_6+b+k_7*x_2^2$
		代码	`lm.new<-update(sol.lm,.~.+I(x2^2))` 它等同于使用 lm 直接建立的模型： `lm(y~x1+x2+x3+x4+x5+x6+I(x2^2),data.train)`

（续）

参数	说明		
Formula	2	修正模型	在模型中删除 $x2$ 变量： $y=k_1*x_1+k_3*x_3+k_4*x_4+k_5*x_5+k_6*x_6+b$
		代码	`lm.new<-update(sol.lm, .~.-x2)` 它等同于使用 lm 直接建立的模型： `lm(y~x1+x3+x4+x5+x6,data.train)`
	3	修正模型	在模型中把 $x2$ 变为 $x2$ 平方变量： $y=k_1*x_1+k_2*x_2^2+k_3*x_3+k_4*x_4+k_5*x_5+k_6*x_6+b$
		代码	`lm.new<-update(sol.lm, y~x1+x3+x4+x5+x6+I(x2^2))` 或 `lm.new<-update(sol.lm, .~. -x2 +I(x2^2))` 它等同于使用 lm 直接建立的模型： `lm(y~x1+I(x2^2)+x3+x4+x5+x6,data.train)`
	4	修正模型	在模型中增加 $x1$ 和 $x2$ 的交互作用变量 $x1*x2$： $y=k_1*x_1+k_2*x_2+k_3*x_3+k_4*x_4+k_5*x_5+k_6*x_6+b+k_7*x_1*x_2$
		代码	`lm.new<-update(sol.lm, .~.+x1*x2)` 它等同于使用 lm 直接建立的模型： `lm(y~.+x1*x2,data.train)`
	5	修正模型	在模型中对 y 开方建模： $y^{1/2}=k_1*x_1+k_2*x_2+k_3*x_3+k_4*x_4+k_5*x_5+k_6*x_6+b$
		代码	`lm.new<-update(sol.lm, sqrt(.)~.)` 它等同于使用 lm 直接建立的模型： `lm(sqrt(y)~.,data.train)`

在 6.2.2 节中可观察到 $x2$ 和 $x4$ 的参数 $k2$ 和 $k4$ 并未通过 T 检验，使用 update 函数修正得到 lm.new，同时观察 T 检验的 p.value 数据 summary(lm.new)$coefficients[,4]，修正判定系数 summary(lm.new)$adj.r.squared 以及残差的标准误差 summary(lm.new)$sigm。使用 updata 函数优化模型的示例如表 6-11 所示。

表 6-11 使用 updata 函数优化模型示例

step1	想法	首先尝试添加 $x2$ 平方和 $x4$ 平方变量对原有模型进行修正		
	代码	`lm.new<-update(sol.lm,.~.+I(x4^2)+I(x2^2))`		
	输出	T 检验的 p.value	修正判定系数	残差的标准误
		`Coefficients:` `Estimate Std. Error t value Pr(>\|t\|)` `(Intercept) -1.194e+02 2.920e+01 -4.088 6.18e-05 ***` `x1 7.811e-01 4.627e-02 16.883 < 2e-16 ***` `x2 3.399e+00 2.754e-01 12.341 < 2e-16 ***` `x3 6.861e-02 4.603e-02 1.491 0.138` `x4 5.241e-01 7.656e-01 0.684 0.494` `x5 3.978e-01 4.224e-02 9.419 < 2e-16 ***` `x6 -2.036e-01 4.633e-02 -4.395 1.76e-05 ***` `I(x4^2) -5.549e-03 5.537e-03 -1.002 0.317` `I(x2^2) -2.862e-02 2.270e-03 -12.606 < 2e-16 ***`	0.7985246	4.686005
	结论	$x3$ 变量、$x4$ 平方变量和 $x4$ 变量均未通过 T 检验		

（续）

<table>
<tr><td rowspan="4">step2</td><td>想法</td><td colspan="3">由 step1 的结论，尝试只添加 x2 平方变量对原有模型 sol.lm 进行修正</td></tr>
<tr><td>代码</td><td colspan="3">lm.new1<-update(sol.lm,.~.+I(x2^2))</td></tr>
<tr><td rowspan="2">输出</td><td>T 检验的 p.value</td><td>修正判定系数</td><td>残差的标准误</td></tr>
<tr><td>Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) -91.544879 9.004634 -10.166 < 2e-16 ***
x1 0.781848 0.046261 16.901 < 2e-16 ***
x2 3.344612 0.270017 12.387 < 2e-16 ***
x3 0.067996 0.046022 1.477 0.141
x4 -0.240845 0.059922 -4.019 8.11e-05 ***
x5 0.397634 0.042240 9.414 < 2e-16 ***
x6 -0.208677 0.046056 -4.531 9.80e-06 ***
I(x2^2) -0.028181 0.002228 -12.648 < 2e-16 ***</td><td>0.7985206</td><td>4.686052</td></tr>
<tr><td>结论</td><td colspan="3">除了 x3 的 T 检验未通过外，其他参数的检验效果均显著提高。该模型虽然不可用，但是可以在它的基础上做二次修正</td></tr>
<tr><td rowspan="4">step3</td><td>想法</td><td colspan="3">由 step2 的结论，可以考虑在 lm.new1 的基础上添加 x3 平方变量进行模型修正</td></tr>
<tr><td>代码</td><td colspan="3">lm.new2<-update(lm.new1,.~.+I(x3^2))</td></tr>
<tr><td rowspan="2">输出</td><td>T 检验的 p.value</td><td>修正判定系数</td><td>残差的标准误</td></tr>
<tr><td>Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) -48.526177 11.801749 -4.112 5.62e-05 ***
x1 0.751359 0.043998 17.077 < 2e-16 ***
x2 3.696168 0.263227 14.042 < 2e-16 ***
x3 -1.879048 0.373599 -5.030 1.05e-06 ***
x4 -0.126574 0.060544 -2.091 0.0378 *
x5 0.402240 0.039832 10.099 < 2e-16 ***
x6 -0.266825 0.044811 -5.954 1.08e-08 ***
I(x2^2) -0.030900 0.002164 -14.282 < 2e-16 ***
I(x3^2) 0.015968 0.003043 5.247 3.75e-07 ***</td><td>0.8209311</td><td>4.417756</td></tr>
<tr><td>结论</td><td colspan="3">所有自变量均通过了 T 检验 (x4 变量效果较差)，同时修正判定系数 (0.8209311) 大于原模型 sol.lm (0.6481513), 残差的标准误 (4.417756) 小于原模型 sol.lm (6.192552)，即修改后的模型不仅可用，而且性能也有所提高</td></tr>
<tr><td rowspan="4">step4</td><td>想法</td><td colspan="3">由 step3 的结论，可以再次考虑在模型 lm.new2 中添加 x4 平方进行修正</td></tr>
<tr><td>代码</td><td colspan="3">lm.new3<-update(lm.new2, .~.+I(x4^2))</td></tr>
<tr><td rowspan="2">输出</td><td>T 检验的 p.value</td><td>修正判定系数</td><td>残差的标准误</td></tr>
<tr><td>Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.469e+02 2.690e+01 -5.460 1.34e-07 ***
x1 7.367e-01 4.264e-02 17.277 < 2e-16 ***
x2 4.055e+00 2.693e-01 15.059 < 2e-16 ***
x3 -2.621e+00 4.049e-01 -6.473 6.65e-10 ***
x4 3.072e+00 7.945e-01 3.866 0.000148 ***
x5 4.049e-01 3.847e-02 10.525 < 2e-16 ***
x6 -2.682e-01 4.327e-02 -6.197 2.99e-09 ***
I(x2^2) -3.374e-02 2.204e-03 -15.306 < 2e-16 ***
I(x3^2) 2.208e-02 3.305e-03 6.679 2.12e-10 ***
I(x4^2) -2.288e-02 5.670e-03 -4.036 7.61e-05 ***</td><td>0.833031</td><td>4.265889</td></tr>
<tr><td>结论</td><td colspan="3">所有自变量均通过了T检验，且效果很好(p.value 均小于 0.001)。同时修正判定系数 (0.833031) 大于上次修正模型 lm.new2 (0.8209311), 残差的标准误 (4.265889) 小于上次修正模型 lm.new2 (4.417756)，即修改后的模型 lm.new3 比 lm.new2 的性能好</td></tr>
<tr><td rowspan="2">step5</td><td>想法</td><td colspan="3">由 step4 的结论可见，变量 x4 的 T 检验效果最差（p.value 数值最大），因此还可以进一步考虑 x4 与其他变量的交互作用，即在 lm.new3 基础上添加 x4 × xi 进行模型修正。这里依次添加 x4 × x1、x4 × x2、x4 × x3、x4 × x5、x4 × x6 变量，观察到添加 x4 × x2 后的修正判定系数最大，同时残差的标准误也很小。因此这里以添加 x4 × x2 变量为例展示代码</td></tr>
<tr><td>代码</td><td colspan="3">lm.new4<-update(lm.new3,.~.+ x4*x2)</td></tr>
</table>

（续）

step5	输出	T 检验的 p.value	修正判定系数	残差的标准误
		Coefficients: Estimate Std. Error t value Pr(>\|t\|) (Intercept) -2.143e+02 2.794e+01 -7.670 6.43e-13 *** x1 7.169e-01 4.005e-02 17.899 < 2e-16 *** x2 5.607e+00 3.761e-01 14.909 < 2e-16 *** x3 -2.939e+00 3.831e-01 -7.671 6.39e-13 *** x4 4.001e+00 7.620e-01 5.251 3.72e-07 *** x5 4.334e-01 3.635e-02 11.921 < 2e-16 *** x6 -3.021e-01 4.094e-02 -7.379 3.71e-12 *** I(x2^2) -3.503e-02 2.075e-03 -16.879 < 2e-16 *** I(x3^2) 2.476e-02 3.130e-03 7.910 1.48e-13 *** I(x4^2) -2.043e-02 5.323e-03 -3.838 0.000164 *** x2:x4 -2.096e-02 3.771e-03 -5.558 8.25e-08 ***	0.8538385 ,	3.991244
	结论	所有自变量均通过了 T 检验 (*x*1 与 *x*4 的交互作用变量 *x*1 × *x*4 效果较差)，同时修正判定系数 (0.8538385) 大于上次的修正模型 lm.new3 (0.833031), 残差的标准误 (3.991244) 小于上次的修正模型 lm.new3 (4.265889)，即修改后的模型 lm.new4 比 lm.new3 的性能好		

表 6-11 最后确定回归模型的 formula 为：

```
summary(lm.new4)$call
lm(formula = y ~ x1 + x2 + x3 + x4 + x5 + x6 + I(x2^2) + I(x3^2) +
    I(x4^2) + x2:x4, data = data.train)
```

即 y ～ x1+x2+x3+x4+x5+x6+I(x2^2)+I(x3^2)+I(x4^2)+x2:x4。注意，这里出现了 x2^2、x3^2、x2 × x4，表示已经不仅仅是线性模型了。

6.2.4 逐步回归分析函数 step()

上一节的例子中使用了 update 函数添加新变量来修正模型并提高其性能。这里介绍一种逐步减少变量的方法——使用 step 函数，使修正后所有模型参数的 T 检验效果尽量提高。step 函数的使用方法如下：

```
lm.step<-step(sol.lm)
```

下面对 lm 函数得到的模型结果 sol.lm（formula 是 $y=k_1 \times x_1+k_2 \times x_2+ k_3 \times x_3+k_4 \times x_4+k_5 \times x_5+k_6 \times x_6+b$）进行逐步回归分析。

1）由系统计算不删除任何变量时，模型的 AIC 数值 (图 6-3 中的 Start AIC)，AIC 是逐步回归效果衡量指标，其数学原理这里不详细介绍，读者只要知道 AIC 越小，模型越好。

2）计算删除模型中某一变量后的 AIC 值（图 6-3 中的 AIC 列），并依据该值由小到大进行排列展示。

由上面结果可知，未删除任何变量时的 AIC 等于 809.16。删除 *x*4 时得到的 AIC 最小，其值等于 807.71。其次是删除 *x*2 时得到的 AIC，其值等于 808.56。删除 *x*1 时得到的 AIC 最大，其值等于 892.34。上面自变量的排列顺序为 *x*4、*x*2、*x*5、*x*6、*x*3、*x*1，这也表示了不同自变量的 T 检验效果，与 sol.lm 函数产生的结果一致，如

```
Start:  AIC=809.16
y ~ x1 + x2 + x3 + x4 + x5 + x6

       Df Sum of Sq     RSS     AIC
- x4    1      20.4  8188.4 807.71
- x2    1      52.2  8220.3 808.56
<none>               8168.1 809.16
- x5    1     223.8  8391.9 813.11
- x6    1     478.2  8646.2 819.68
- x3    1    1583.2  9751.2 846.13
- x1    1    3862.1 12030.2 892.34
```

图 6-3 删除某一变量后的 AIC 值

图 6-4 所示。

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 14.60962    4.31039   3.389 0.000835 ***
x1           0.57316    0.05711  10.036  < 2e-16 ***
x2          -0.04787    0.04103  -1.167 0.244564
x3           0.34403    0.05354   6.425 8.47e-10 ***
x4          -0.05600    0.07680  -0.729 0.466704
x5           0.11433    0.04733   2.416 0.016542 *
x6          -0.21491    0.06086  -3.531 0.000507 ***
```

图 6-4 结果示例

由上面 sol.lm 函数的结果可知，$x4$ 的效果最差（其 p.value 最大），其次是 $x2$，$x1$ 的 T 检验效果最好（其 p.value 最小）。

3）系统删除变量 $x4$（上一步删除 $x4$ 变量后得到的 AIC 最小），并计算此时的 AIC 数据，以及在此基础上删除某一剩余变量后得到的 AIC 数值，并由小到大进行排列展示，结果如图 6-5 所示。

可见删除 $x2$ 变量后得到的 AIC 最小，然后依次是 $x5$、$x6$、$x3$、$x1$。

以此类推，接下来删除变量 $x2$，并计算相关的 AIC 数据，展示结果如图 6-6 所示。

```
Step:  AIC=807.71
y ~ x1 + x2 + x3 + x5 + x6

       Df Sum of Sq     RSS    AIC
- x2    1      39.8  8228.3 806.77
<none>               8188.4 807.71
- x5    1     204.5  8393.0 811.13
- x6    1     550.9  8739.3 820.03
- x3    1    1579.7  9768.1 844.52
- x1    1    4760.3 12948.8 906.53
```

图 6-5 结果示例

```
Step:  AIC=806.77
y ~ x1 + x3 + x5 + x6

       Df Sum of Sq     RSS    AIC
<none>               8228.3 806.77
- x5    1     192.8  8421.1 809.87
- x6    1     552.8  8781.0 819.08
- x3    1    1542.9  9771.1 842.58
- x1    1    4995.4 13223.7 909.15
```

图 6-6 结果示例

可以发现，此后无论删除哪个变量，其 AIC 数值均大于初始状态的 AIC 数值 806.77（所有变量均处在 <none> 以下），因此系统停止删除任何变量，得到逐步回归后的模型。

观察 lm.step 的模型参数结果：

```
summary(lm.step)
```

其中各参数的 T 检验 p.value 数值如图 6-7 所示。

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 13.23521    4.12268   3.210 0.001529 **
x1           0.53637    0.04695  11.425  < 2e-16 ***
x3           0.32951    0.05190   6.349 1.27e-09 ***
x5           0.09847    0.04387   2.245 0.025811 *
x6          -0.22520    0.05926  -3.800 0.000188 ***
```

图 6-7 优化后的参数结果

可见 $x5$ 变量虽然通过了 T 检验，但是其效果较差，使用 6.2.3 节的 update 函数对模型添加 $x5$ 平方变量，得到如下修正模型：

```
lm.new<- update(lm.step,.~.+I(x5^2))
summary(lm.new)
```

产生的结果如图 6-8 所示。

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.007e+02  1.367e+01  -7.367 3.73e-12 ***
x1           4.938e-01  4.083e-02  12.096  < 2e-16 ***
x3           3.187e-01  4.482e-02   7.112 1.69e-11 ***
x5           3.406e+00  3.851e-01   8.847 3.42e-16 ***
x6          -2.454e-01  5.121e-02  -4.792 3.09e-06 ***
I(x5^2)     -2.251e-02  2.607e-03  -8.633 1.38e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.34 on 214 degrees of freedom
Multiple R-squared:  0.7443,    Adjusted R-squared:  0.7383
F-statistic: 124.6 on 5 and 214 DF,  p-value: < 2.2e-16
```

图 6-8 优化后的参数结果

可见，所有参数的 T 检验效果均不错（标注了三颗星，即小于 0.001），该模型的性能虽然没有 6.2.3 节的模型好（残差标准误较大，修正判定系数也较小），但是模型较为简单，可读性强。

6.2.5 自变量中包含分类型数据的回归分析

以上介绍了自变量均是连续变量时的回归分析，实质上自变量也可以包含个别分类型变量，甚至所有的自变量均可以是分类型变量。对于一个取值为 1，2，…，p 的不同水平的分类型变量 a，首先将其转换为 p 个二元变量 $a1$，$a2$，…，ap，每个变量表示变量 a 的取值是否为 i（i 取 1，2，…，p）。然后对其进行回归分析，产生 $f(a1,a2,..,ap)=y$ 的模型，如果分类变量 a 的取值为 i，则模型预测值是：

$$f(a1=0, a2=0, ..ai=1, .., ap=0)$$

其中，除了 ai 等于 1 外，其他均取值为 0。

这里使用第 5 章中的月销售数据 sales.data 为例，研究不同季度（分类变量）和月份（连续变量，相当于时间序列分析中的时间轴）对销售数据的模型，其中 sales.data 是一个向量数据，代表 2006 年至 2012 年每月的销售数据。使用 lm 函数对其进行分析，代码如下：

```
x.t<-c(1:length(sales.data))
x.season<-rep(gl(4,3),7)
data<-data.frame(sales=sales.data,t=x.t,season=x.season)
sol.lm<-lm(sales~.,data=data)
```

x.t 是月份序列号，其为连续型自变量。x.season 代表每年不同季度的数据，其为取值是 1、2、3、4 的分类变量。使用数据框 data 来统一包含以上自变量数据和销售变量（目标变量）数据。使用 lm 函数对其进行建模（$k1 \times season1+\cdots+k4 \times season4+k5 \times t+b=sales$），并进行回归分析，season1~ season 4 是由 season 转化的二元分类变量，表示 season 是否等于 1~4。结

果如下：

```
summary(sol.lm)
Call:
lm(formula = sales ~ ., data = data)
<中间省略>
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -3032.14    3283.90  -0.923 0.048644   *
t             333.41      53.62   6.218 2.23e-08 ***
season2      2648.38    3645.58   0.726 0.069702   *
season3     14843.05    3656.21   4.060 0.000115 ***
season4     -1426.72    3673.86  -0.388 0.028807   *
---
Signif. codes:  0‘***’ 0.001‘**’ 0.01‘*’ 0.05‘.’ 0.1‘ ’ 1
<以后省略>
```

根据结果确定最后的模型为：

333.41 × t+2648.38 × season2+14843.05 × season3-1426.72 × season4-3032.14=sales

以 2006 年 8 月为例，此时 t=8，season=3（第 3 季度），则对应模型中的变量为 t=8，season1=0，season2=0，season3=1，season4=0。此时预测的模型数据为 333.41 × 8+14843.05 –3032.14= 14478.19。使用下面的语句来统一绘制原有数据（黑色曲线）和模型预测数据（红色曲线）的曲线图。

```
plot(sales.data,type="l")
lines(predict(sol.lm),col="red",type="l")
```

绘制结果如图 6-9 所示。

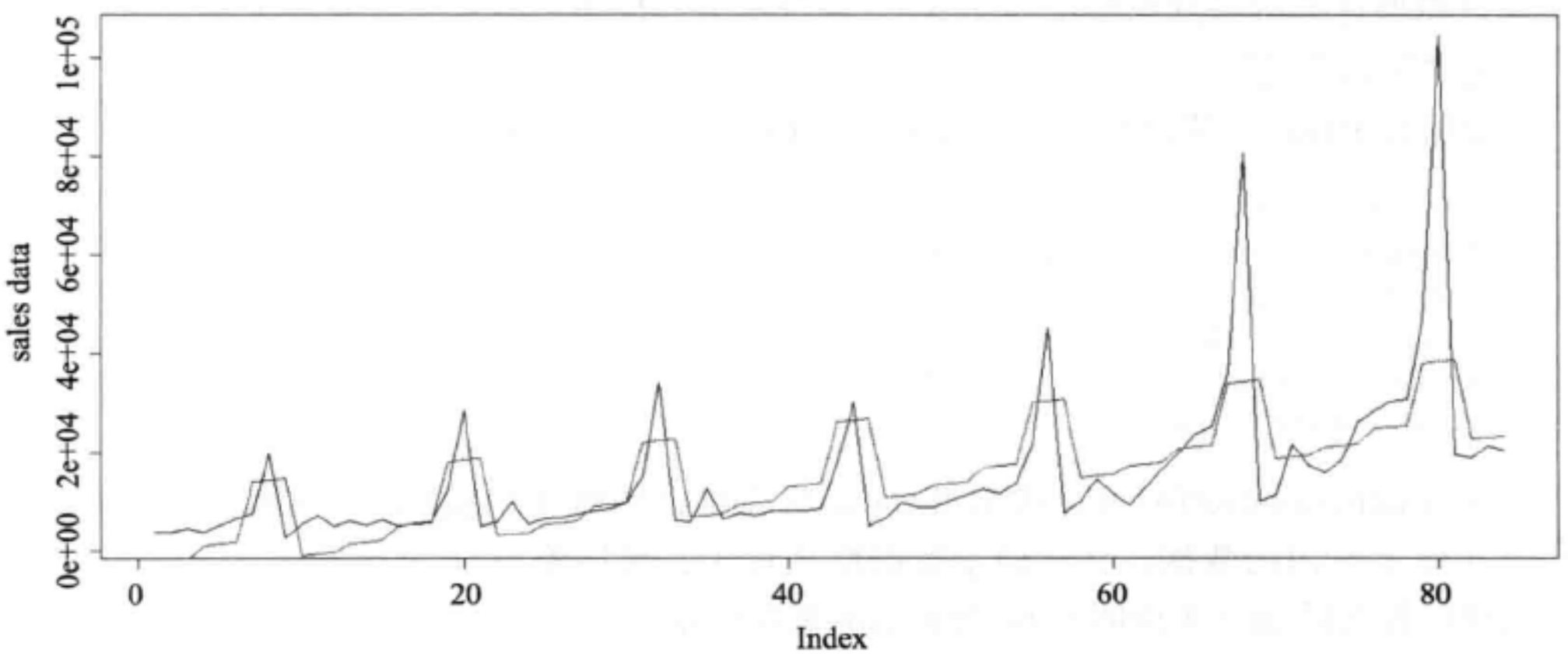


图 6-9 原有数据和模型预测数据的曲线

也可以使用 t 的平方变量来建立 sales 的模型。

```
sol.lm<-lm(sales~t+I(t^2)+season,data=data)
```

具体过程这里不再详细介绍，读者可以综合参考前几章的内容。

6.3 Logic 回归分析

理论上，回归分析是在目标变量为连续型数据的情况下建模的，它并不能处理目标变量为分类型数据的情况。例如，想通过部分用户行为数据研究用户是否会另付费开通 VIP 会员（分类变量：购买为 1，不购买是 0），实际是第 7 章分类分析的研究范畴。而 logic 回归分析的思路是把“是否会开通 VIP 会员”（分类变量）转化为“开通 VIP 会员的概率”（连续变量），进而使用回归分析的方法间接地研究分类分析的问题。

6.3.1 引例及数据

data.xls 文件是某购物网站 4000 名用户的行为数据，其中包含的变量如表 6-12 所示。

表 6-12 data.xls 文件包含的变量

名称	含义
gender	用户性别（分类型变量），0= 男，1= 女
m	用户在网站上累计消费的总金额
r	距离最后一次购买的月数
f	总的购买次数
register	用户注册后至今的月数
vip	是否付费开通了 VIP 会员（分类变量），0= 未开通，1= 已开通

在使用 logic 分析来研究用户愿意付费开通 VIP 会员的概率（即 vip=1 的概率）时，如果通过模型计算后得到的概率大于 0.5，则预测为“该用户愿意付费开通 VIP 会员”，否则预测为“不愿开通”。

使用 RODBC 包读取 C:/r_lab/ 目录下的 Excel 数据文件 data.xls，代码如下：

```
root<-"C:/r_lab/"
file<-paste(root,"data.xls",sep="")
library (RODBC)
excel_file <- odbcConnectExcel(file)
data<- sqlFetch ( excel_file,"data")
close ( excel_file )
```

通过 odbcConnectExcel 函数打开 data.xls 文件，然后使用 sqlFetch 函数读取 data.xls 文件中 data 子页内的数据，并存储于数据框 data 中，最后使用 close 函数关闭 data.xls 文件。下面使用数据框 data 来说明 logic 分析的原理和应用。

6.3.2 logic 分析的原理

1. 概述

（1）目标变量的 logic 转化

由于 vip 变量是分类变量，其取值只有 0 有 1 两种，所以无法使用回归分析得到

$$\text{vip}=k_1\times\text{gender}+k_2\times m+k_3\times r+k_4\times f+k_5\times\text{register}+b$$

形式的模型。但是在不同的取值（如 gender、m、r、f、register）下，vip 取值为 1 的概率却是一个连续变量（称为 prob.vip），这时可以使用回归分析为 prob.vip 建模：

$$\text{prob.vip}=k_1\times\text{gender}+k_2\times m+k_3\times r+k_4\times f+k_5\times\text{register}+b$$

由于 $k_1\times\text{gender}+k_2\times m+k_3\times r+k_4\times f+k_5\times\text{register}+b$ 的取值范围是 [$-\infty$,$+\infty$]，而 prob.vip 的范围应该是 [0,1]，因此，使用 y=1/(1+exp(-x)) 函数进行转化，可以使取值范围均统一为 [0,1]。y=1/(1+exp(-x)) 的函数曲线如图 6-10 所示。

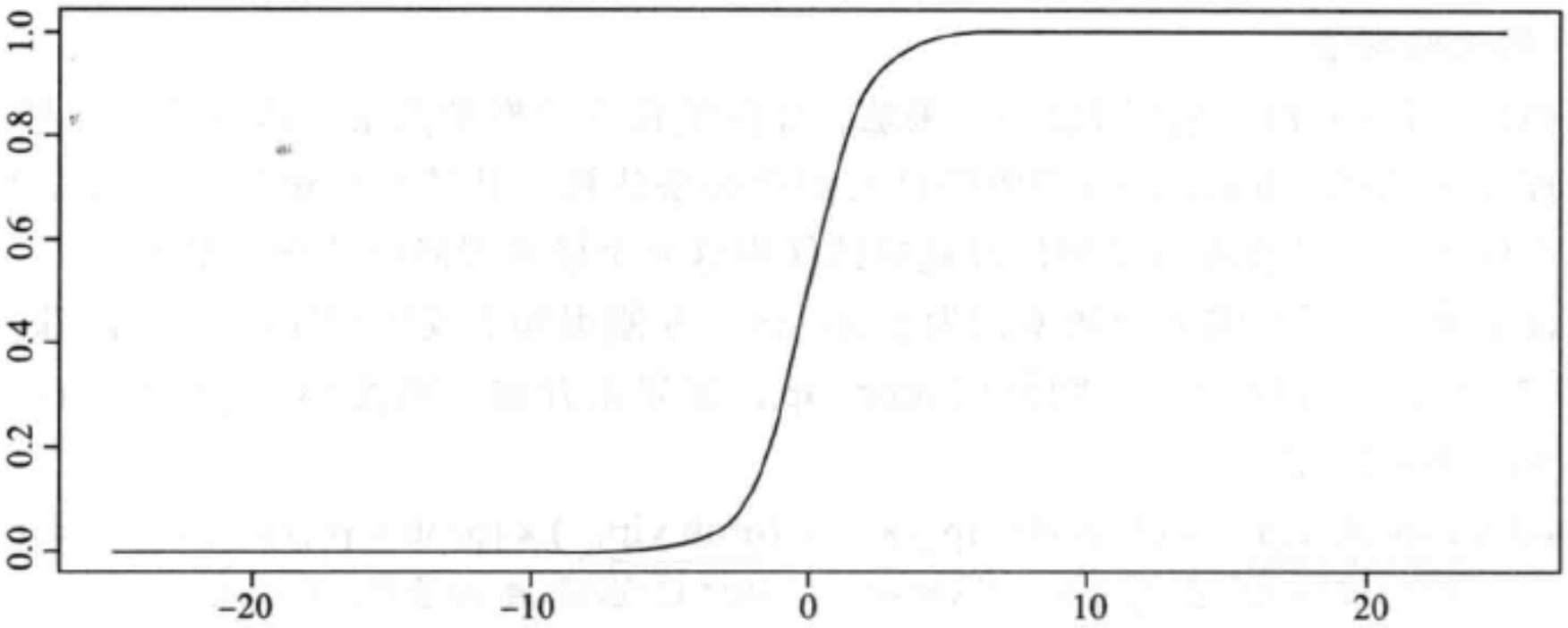


图 6-10　1/(1+exp(-x)) 函数曲线

通常，把 $k_1\times\text{gender}+k_2\times m+k_3\times r+k_4\times f+k_5\times\text{register}+b$ 称为线性拟合模型，记为 linear.vip。并把 linear.vip 带入函数 y=1/(1+exp(-x)) 中，最后修改为：

$$\text{prob.vip}=\frac{1}{1+e^{-(\text{linear.vip})}}=\frac{1}{1+e^{-(k_1*\text{gender}+k_2*m+k_3*r+k_4*f+k_5*\text{register}+b)}}$$

并称为 logic 拟合模型。

（2）目标变量的模型预测以及模型性能衡量

在使用 logic 拟合模型计算 vip=1 时的概率（prob.vip）后，如果 prob.vip 大于 0.5，则可以预测目标变量 vip.predict=1，否则 vip.predict=0。

虽然 logic 回归分析也有判定系数的概念，但是它并不能很好地衡量模型性能，这里使用模型的预测正确率来衡量。目标变量的实际样本数值和预测数值间的关系如表 6-13 所示。

表 6-13　实际数据和预测数据关系

		预测的 vip 变量数据：vip.predict	
		1	0
样本中 vip 变量数据：vip	1	num_{11}	num_{10}
	0	num_{01}	num_{00}

通过表 6-13 可得到如下公式：

预测正确率 $=(num_{11}+num_{00})$/ 样本总数量 $=(num_{11}+num_{00})/(num_{11}+num_{10}+num_{01}+num_{00})$

该数值越接近 1，性能越好。



注意

由于 gender 是分类变量，所以实际上要把其分为“gender 是否等于 1”和“gender 是否等于 0”这两个二元变量 (分别是 gender1 和 gender0) 来处理，即线性拟合模型 linear.vip= $k_1 \times$ gender1+ $k_2 \times$ gender0+$k_3 \times$ m+$k_4 \times r$+$k_5 \times f$+$k_6 \times$ register+b。同时，linear.vip 也不单单限于各自变量的线性组合，它也可以是各自变量的 n 次方或不同自变量间乘积的线性组合。

2. 最大似然法

前面介绍回归分析是使用最小二乘法（旨在使获得的模型能最大限度地拟合样本数据）来估算模型参数的，logic 分析则使用最大似然法来估算。其基本思想是：当从总体中随机抽取 n 个样本后，最合理的参数估计量应该使得这 n 个样本观测值的概率最大。

假设把第 i 个用户的开通概率记为 $prob.vip_i$，并使用如下规则构建出似然量 likelihood：如果用户 i 开通了 VIP 会员，则使用 $prob.vip_i$，如果未开通，则使用 (1-prob.vipi)，进行连乘构成 likelihood，即

$$likelihood=(\underbrace{1-prob.vip_1}_{\text{用户 1 实际未开通 vip 会员，即 vip1=0}}) \times (1-prob.vip_2) \times \ldots \times (\underbrace{prob.vip_{12}}_{\text{用户 12 实际开通 vip 会员，即 vip12=1}}) \times (prob.vip_{13}) \times \ldots \times (1-prob.vip_{4000})$$

并定义对数似然量 log.likelihood=likelihood，可通过 log.likelihood 取值最高这一条件来估算出模型 $prob.vip=\frac{1}{1+e^{-(k_1*gender+k_2*m+k_3*r+k_4*f+k_5*register+b)}}$ 中的参数 k 和 b。

6.3.3 R 语言实现

1. 观察数据间的关系

首先观察不同自变量与目标变量 vip 间是否存在显著的差异性。由于 gender 是分类变量，因此这里使用列联表分析：

```
vip.gender<-table(vip.data$vip,vip.data$gender)
summary(vip.gender)
  Number of cases in table: 4000
  Number of factors: 2
  Test for independence of all factors:
          Chisq = 17.98, df = 1, p-value = 2.233e-05
```

从上述结果可知，通过卡方分析得到的 p.value 为 2.233e-05，远小于 0.05，那么可以认为 vip 和 gender 变量有显著性差异。通过如下语句绘制柱状图。

```
barplot(round(t(t(vip.gender)/apply(vip.gender,2,sum)),2),xlab="gender")
```

绘制结果如图 6-11 所示。

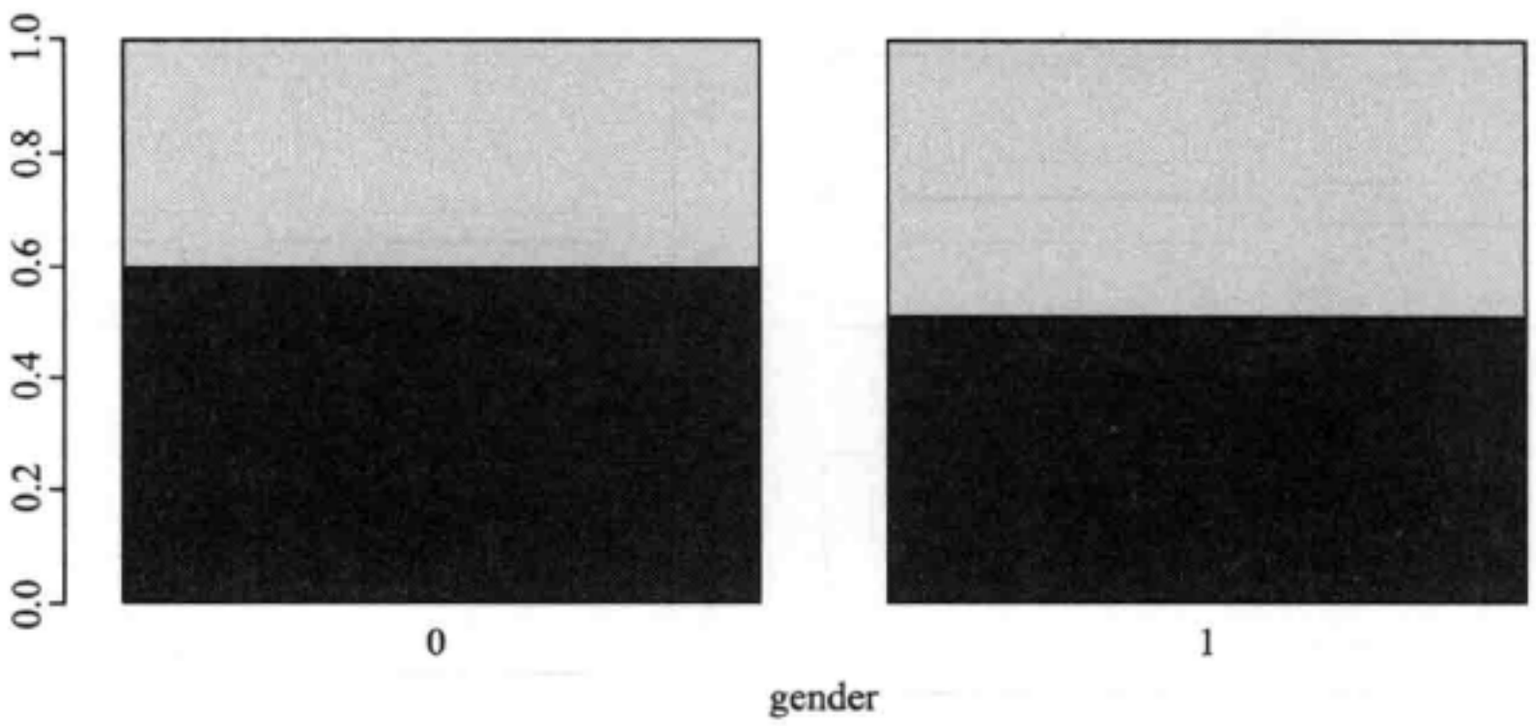


图 6-11 列联表分析柱状图

可见，gender 在取 0 和 1 时，vip 取 0 和 1 的比例明显不同，即 gender 很可能对 vip 的取值有影响。

因为自变量 m、r、f 以及 register 是连续型数据，所以使用 T 检验来衡量它与 vip 的显著关系。代码如下：

```
    t.test(vip.data$m[which(vip.data$vip==1)],vip.data$m[which(vip.
data$vip==0)])$p.value
    [1] 2.878091e-31
    t.test(vip.data$r[which(vip.data$vip==1)],vip.data$r[which(vip.
data$vip==0)])$p.value
    [1] 1.246573e-09
    t.test(vip.data$f[which(vip.data$vip==1)],vip.data$f[which(vip.
data$vip==0)])$p.value
    [1] 5.296754e-20
    t.test(vip.data$register[which(vip.data$vip==1)],vip.data$register[which(vip.
data$vip==0)])$p.value
    [1] 2.887918e-28
```

最后的结果显示 T 检验的 p.value 数据均远小于 0.05，即 vip 与不同自变量间均有显著差异。可通过如下语句分别绘制 vip 与 m、r、f 以及 register 的线箱图。

```
par(mfrow=c(1,4))
plot(vip.data$vip,vip.data$m,main="vip-m")
plot(vip.data$vip,vip.data$r,main="vip-r")
plot(vip.data$vip,vip.data$f,main="vip-f")
plot(vip.data$vip,vip.data$register,main="vip-register")
```

绘制结果如图 6-12 所示。

可见 vip 取 0 或 1 时，相应自变量的分布情况也不同。

2. glm 函数

glm() 是用 R 语言实现 logic 回归分析的核心函数，其基本参数说明见表 6-14。

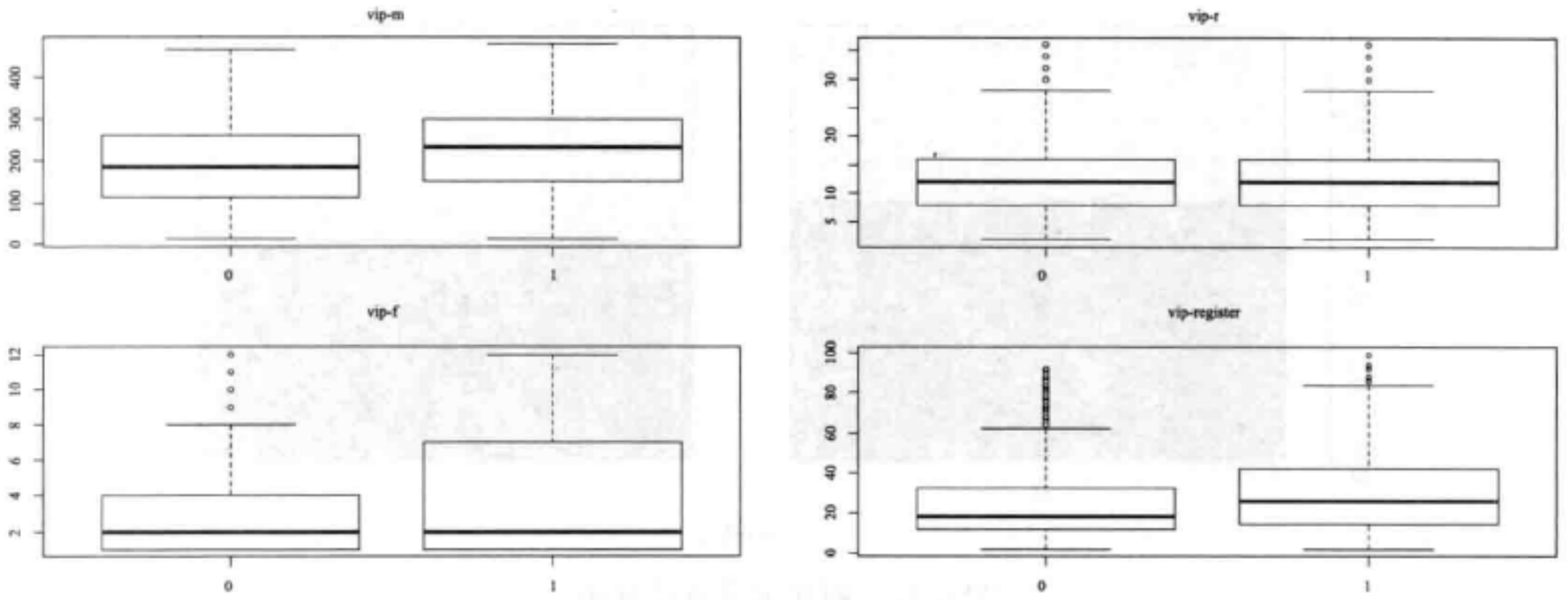


图 6-12 线箱图

表 6-14

参数	说明
formula	设置线性拟合模型 linear.vip 的形式。例如，formula=gender+m+r+f+$I(f$^2) 表示： 线性拟合模型为 linear.vip=$k_1 \times$ gender+$k_2 \times m$+$k_3 \times r$+$k_4 \times f$+$k_5 \times f^2$+b logic 拟合模型为 prob.vip= $\frac{1}{1+e^{-(k_1*\text{gender}+k_2*m+k_3*r+k_4*f+k_5*f^2+b)}}$
family	glm 函数的算法族，对于 logic 回归分析，family 要设置为 binomial("logit")
data	样本数据，一般以数据框形式给出

（1）建立 logic 回归分析模型

下面对 6.3.1 节引例中的数据进行 logic 回归分析。代码如下：

```
sol.glm <- glm(vip~.,data=vip.data,family=binomial("logit"))
summary(sol.glm)
Call:
glm(formula=vip~.,family=binomial("logit"),data=vip.data)
      <中间省略>
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.5018444  0.1115197 -13.467  < 2e-16 ***
gender1      0.3163952  0.0720515   4.391 1.13e-05 ***
m            0.0030494  0.0003783   8.061 7.55e-16 ***
r            0.0147850  0.0067810   2.180   0.0292 *
f           -0.0006333  0.0259416  -0.024   0.9805
register     0.0097463  0.0054079   1.802   0.0715 .
---
Signif. codes: 0 '***'0.001'**'0.01'*'0.05'.'0.1' '1
<以后省略>
```

因为 glm 函数的 formula 参数设置为 vip~.，并且 gender 是分类变量，所以建立的模型

形式是：

$$\text{prob.vip}=\frac{1}{1+e^{-(k_1*\text{gender1}+k_2*\text{gender0}+k_3*m+k_4*r+k_5*f+k_6*\text{register}+b)}}$$

其中，gender1 表示 gender 是否等于 1，gender0 表示 gender 是否等于 0。得到结果中的 Coefficients 部分表示模型参数 k 和 b 的数值和检验结果，其中 f 和 register 没有通过检验。

使用 step 函数对模型进行修正。代码如下：

```
sol.glm<-step(sol.glm)
summary(sol.glm)
  Call:
  glm(formula=vip~gender+m+r+register,family=binomial("logit"),
  <中间省略>
  Coefficients:
               Estimate Std. Error z value Pr(>|z|)
  (Intercept) -1.5023700  0.1094236 -13.730  < 2e-16 ***
  gender1      0.3163974  0.0720515   4.391 1.13e-05 ***
  m            0.0030477  0.0003719   8.196 2.49e-16 ***
  r            0.0149064  0.0046093   3.234  0.00122 **
  register     0.0096264  0.0022607   4.258 2.06e-05 ***
  ---
  Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
  <以后省略>
```

修正后的模型认为 gender、m、r 以及 register 变量可以很好地预测 vip，最后的模型为：

$$\text{prob.vip}=\frac{1}{1+e^{-(0.3163974*\text{gender1}+0.0030477*m+0.0149064*r+0.0096264*\text{register}-1.5023700)}}$$

其中 gender1 表示 gender 是否等于 1。

（2）回归分析模型 glm 的输出项

在上个要点中，sol.glm 是由 glm 产生的 logic 分析模型结果，使用 str 函数查看其包含的数据属性。代码如下：

```
str(sol.glm)
List of 31
 $ coefficients: Named num [1:5] -1.50237 0.3164 0.00305 0.01491 0.00963
  ..- attr(*, "names")= chr [1:5] "(Intercept)" "gender1" "m" "r" ...
 $ residuals: Named num [1:4000] 1.87 3.45 1.9 2.56 2.17 ...
  ..- attr(*, "names")= chr [1:4000] "1" "2" "3" "4" ...
 $ fitted.values: Named num [1:4000] 0.535 0.29 0.525 0.391 0.461 ...
  ..- attr(*, "names")= chr [1:4000] "1" "2" "3" "4" ...
 $ effects: Named num [1:4000] 5.51 3.95 11.27 6.06 4.26 ...
  ..- attr(*, "names")= chr [1:4000] "(Intercept)" "gender1" "m" "r" ...
 <以后省略>
```

使用 str 函数还可以查看 sol.glm 内部包的 31 个更为复杂的数据项，这里针对比较常用

的输出结果加以说明，具体见表 6-15。

表 6-15 常用模型输出结果的说明

输出项	直接读取方式	说明
模型参数	sol.glm$coefficients	通过该数据可以确定线性拟合模型 linear.vip 中的参数 k 和 b
线性拟合模型的预测数据	sol.glm$linear.predictors	样本数据代入线性拟合模型后得到的 linear.vip 数据
vip 等于 1 的概率 prob.vip	sol.glm$fitted.values	sol.glm$linear.predictors 代入 1/(1+exp(-x)) 函数后得到的 prob.vip 数据
线性拟合模型的残差	sol.glm$residuals	由 sol.glm$residuals+sol.glm$linear.predictors 代入 1/(1+exp(-x)) 函数后得到的数据，再经过阈值为 0.5 的单位阶跃函数处理后得到的数值，该值与样本中的 vip 相等

3. 模型预测以及性能衡量

（1）模型预测

模型给出了 vip=1 时的概率 sol.glm$fitted.values，使用阈值是 0.5 的单位阶跃函数处理后可以得到 vip 的预测数值 predict.vip。代码如下：

```
predict.vip<-ifelse(sol.glm$fitted.values>=0.5,1,0)
predict.vip<-as.factor(predict.vip)
```

如果 sol.glm$fitted.values 内元素大于等于 0.5，则对应的 predict.vip 等于 1，否则等于 0。

对于新样本 test.vip.data 的预测可使用 predict 函数，首先计算线性拟合模型的预测值，再通过 1/(1+exp(–x)) 计算 vip=1 时的概率值，最后使用 ifelse 阶跃函数来预测最后数值。代码如下：

```
new.predict.vip<-predict(sol.glm,newdata=test.vip.data)
new.predict.vip<-1/(1+exp(-new.predict.vip))
new.predict.vip<- as.factor(ifelse(new.predict.vip >=0.5,1,0))
```

注意，test.vip.data 必须是含有自变量列名称的数据框。

（2）模型的性能衡量

根据 6.3.2 节中讲解的性能衡量原理来估测，计算模型对训练集 vip.data 的预测正确率为：

```
    sol.performance<-length(which((predict.vip==vip.data$vip)==TRUE))/nrow(vip.
data);sol.performance
    [1] 0.6185
```

其中 length(which((predict.vip==vip.data$vip)==TRUE)) 是预测值 (predict.vip) 与实际样本值 (vip.data$vip) 元素取值相等的数目，nrow(vip.data) 为样本数。模型最终的正确率为 61.85%。

6.4 回归树 CART

回归树（也称分类回归树 CART）主要以一种树状结构来表达回归分析模型的回归算法，该类方法不仅可以应用于回归分析（称为回归树），也可用于分类分析（称为分类树）。

6.4.1 rpart 函数

R 语言实现 CART 算法的核心函数是 rpart 包的 rpart 函数，这里以 6.2.1 节的数据为例，下面使用训练集 data.train 的 $x1$ 和 $x3$ 变量建立目标变量 y 的回归树模型。代码如下：

```
library(rpart)
sol.rpart<-rpart(y~x1+x3,data=data.train)
```

使用 plot 函数绘制树形模型结构图。代码如下：

```
plot(sol.rpart,uniform=TRUE,compress=TRUE,lty=3,branch=0.7)
text(sol.rpart,all=TRUE,digits=7,use.n=TRUE,cex=0.9,xpd=TRUE)
```

结果输出一个有 19 个节点的二分支的回归树，如图 6-13 所示。

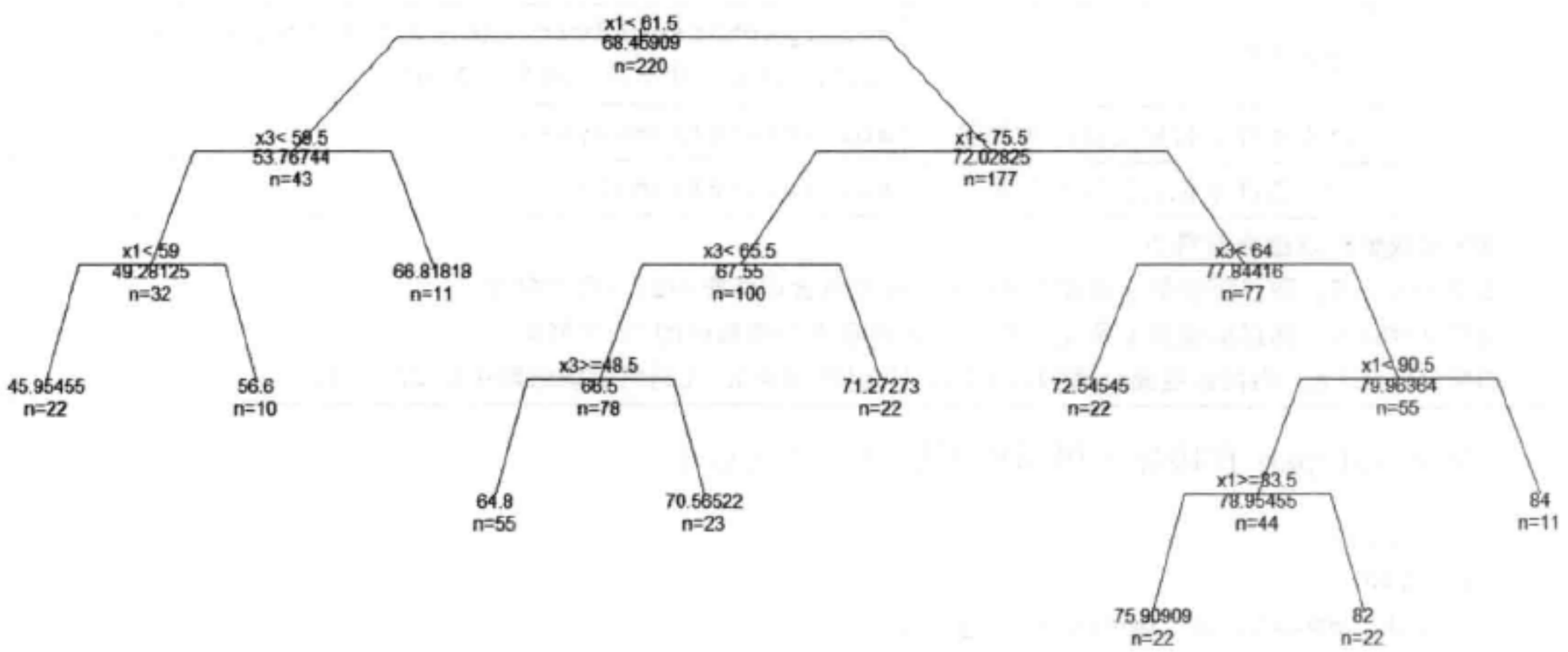


图 6-13 二分支回归树状图

把继续做分支的节点叫决策节点（non-leaf node），不再进行分支的节点叫叶节点（leaf node）。从图 6-13 可看出，该决策树包含 10 个叶节点和 9 个决策节点。接下来以图中 x<65.5 的分支为例来解释图 6-13 中的信息，如图 6-14 所示。

具体分析见表 6-16。

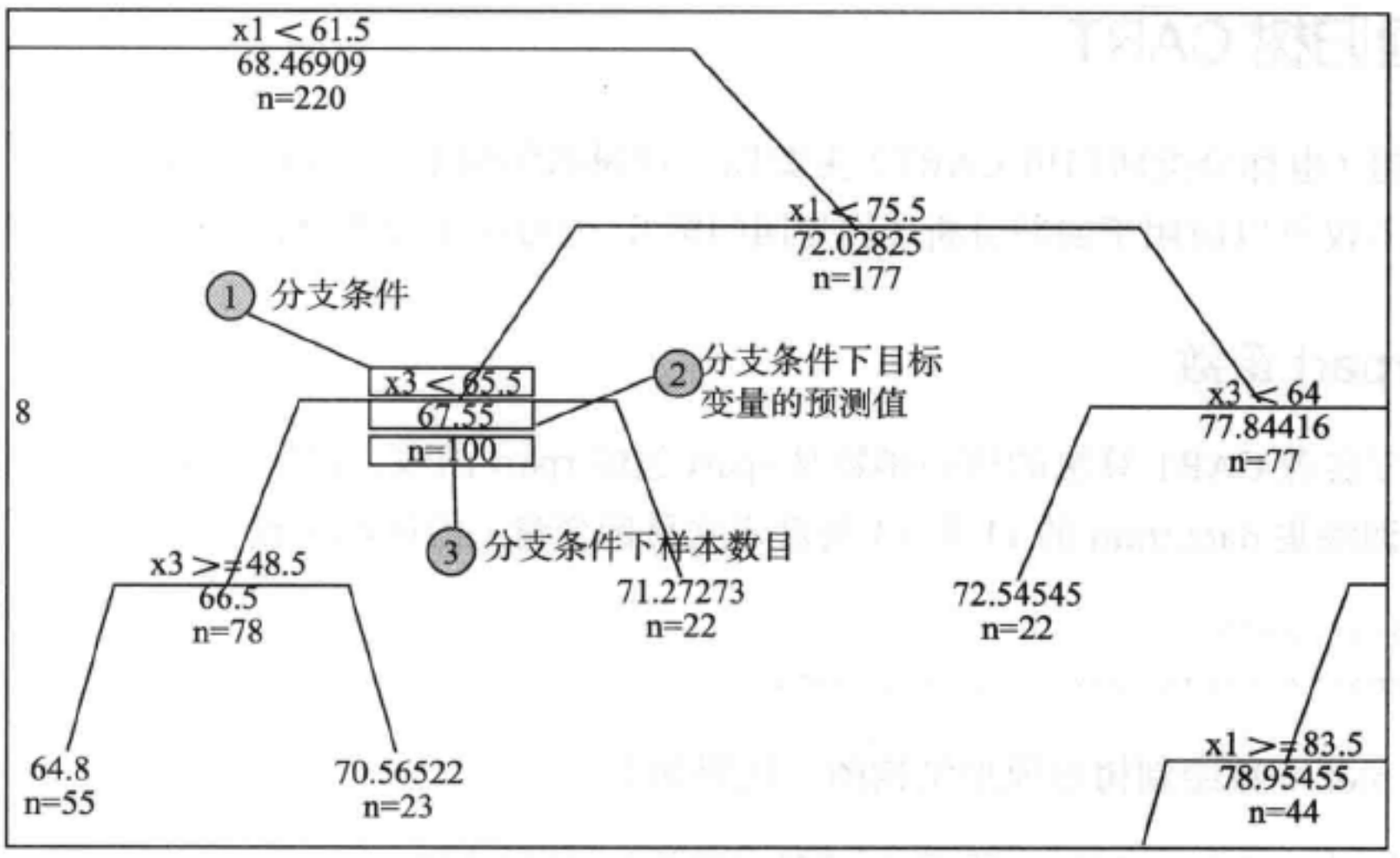


图 6-14 分支详细说明

表 6-16 读取分支内容的代码

图中序号	说明	直接读取的代码
1	分支条件	**sol.rpart$frame$var** 读取分支条件中的变量（名称 "**x1**" 或 "**x3**"），如果是叶节点，则为 "**<leaf>**"
2	分支条件下目标变量的预测量	**sol.rpart$frame$yval**
3	分支条件下包含的样本数量	**sol.rpart$frame$n**

该区域模型可以综合解释为：

如果 *x*1<75.5，则目标变量 *y* 预测为 67.55，此时包含训练集中的 100 个样本。

如果 *x*3<65.5，则目标变量 *y* 预测为 66.5，此时包含训练集中的 78 个样本。

如果 *x*3>=65.5，则目标变量 *y* 预测为 71.27（leaf 叶节点），此时包含训练集中的 22 个样本

输入 sol.rpart 直接显示树形模型信息。代码如下：

```
sol.rpart
n= 220
node), split, n, deviance, yval
      * denotes terminal node
 1) root 220 23868.63000 68.45909
   2) x1< 61.5 43  3895.67400 53.76744
     4) x3< 59.5 32  1358.46900 49.28125
       8) x1< 59 22    14.95455 45.95455 *
       9) x1>=59 10   564.40000 56.60000 *
     5) x3>=59.5 11    19.63636 66.81818 *
   3) x1>=61.5 177  8436.85900 72.02825
     6) x1< 75.5 100  2136.75000 67.55000
      12) x3< 65.5 78  1563.50000 66.50000
```

```
    24) x3>=48.5 55    968.80000 64.80000 *
    25) x3< 48.5 23     55.65217 70.56522 *
  13) x3>=65.5 22    182.36360 71.27273 *
7) x1>=75.5 77  1690.13000 77.84416
  14) x3< 64 22     57.45455 72.54545 *
  15) x3>=64 55    767.92730 79.96364
    30) x1< 90.5 44    531.90910 78.95455
      60) x1>=83.5 22     31.81818 75.90909 *
      61) x1< 83.5 22     92.00000 82.00000 *
    31) x1>=90.5 11     12.00000 84.00000 *
```

结果中，n=220 表明训练集中数据为 220 个样本数，* 号表明该节点是 leaf 叶节点，该模型共包括 10 个叶节点。例如

6）x1< 75.5 100 2136.75000 67.55000

上述语句含义是：节点号为 6（node），节点分支条件是 x1< 75.5（split），此时包括样本数为 100（n），分支后的差异性为 2136.75000（deviance），预测的目标变量 y 等于 67.55000（yval），该节点并非叶节点（无 * 号）。

除了使用 rpart 包中自带的 plot 函数外，还可以使用 maptree 包的 draw.tree 函数绘制更为复杂的树形结构图。代码如下：

```
library(maptree)
draw.tree(sol.rpart)
draw.tree(sol.rpart,nodeinfo=T)
```

如果只想绘制分支条件及叶节点 leaf 中的信息，则使用代码：draw.tree(sol.rpart)。如果还要绘制决策点内的信息，则把参数 nodeinfo 置为 TRUE：draw.tree(sol.rpart,nodeinfo=T)。

图 6-15 是 $x1$ 和 $x3$ 的散点图，给样本点所属不同模型的叶节点添加不同颜色，并绘制出分支条件。

在二维自变量空间内，样本点被分为 10 块不同的方形区域，每块内样本的目标变量被预测为统一的数值。即回归树模型把连续变量 y 最后固定在 10 个不同的取值之中，可以使用 sol.rpart$where 来直接读取 220 个训练集样本的 y 变量所在的叶节点序号。代码如下：

```
leaf<-which(sol.rpart$frame$var=="<leaf>")
for(i in 1:nrow(data.train)){
      point.col[i]<-which(leaf==sol.rpart$where[i])
}
plot(data.train$x1,data.train$x3,pch=16,col=point.col,xlab="x1",ylab="x3")
```

其中，leaf 是所有叶节点的序号，上面代码把 10 个不同区域内的样本点设置为 10 种不同颜色。

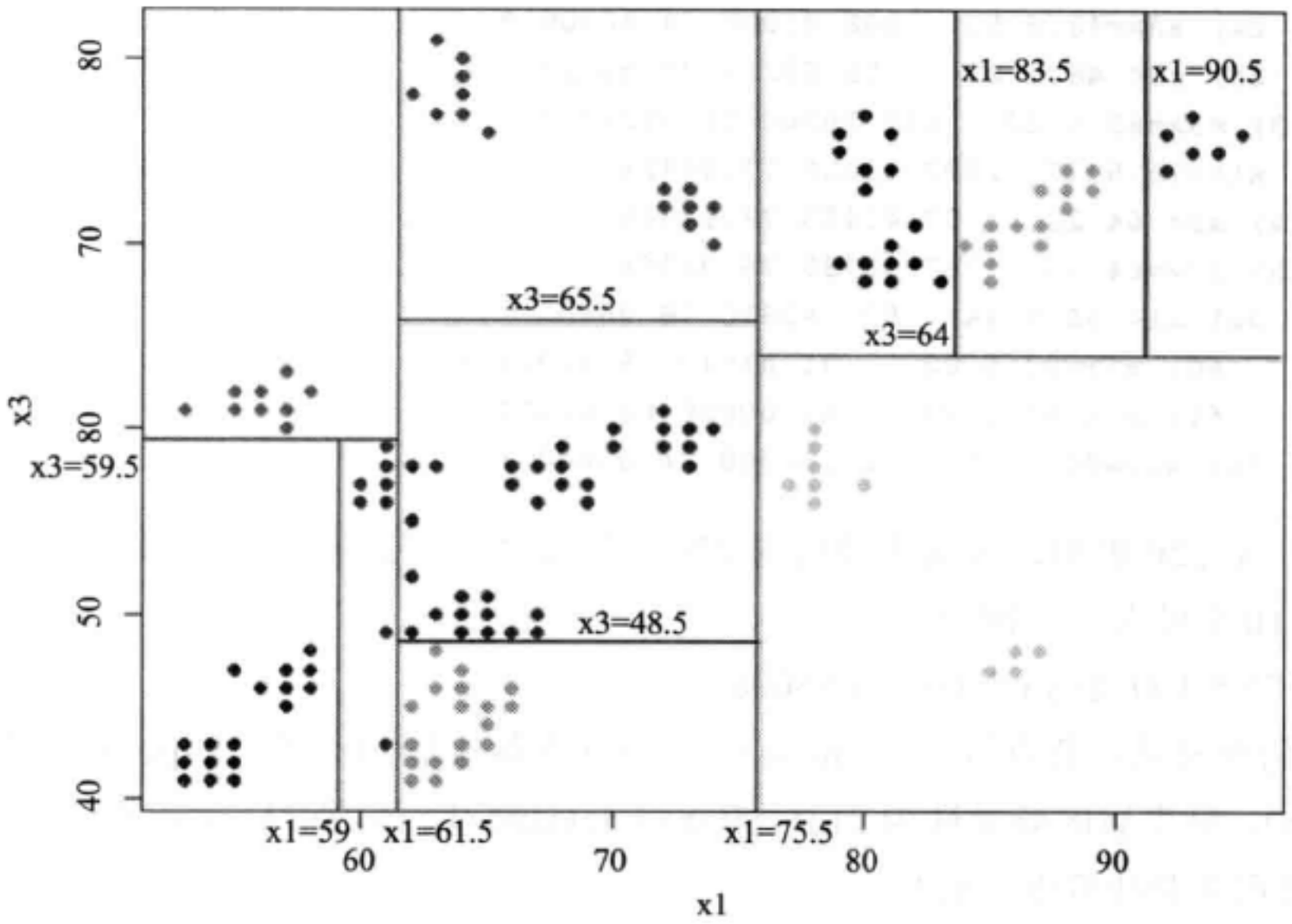


图 6-15 散点分割图

6.4.2 预测及模型性能衡量

使用 predict 函数可以预测模型的样本，并使用所有预测误差（y_i-y.predict$_i$）的总和来衡量模型的性能：

$$\text{sol.error} = \sqrt{\frac{(y_1-y.\text{predict}_1)^2+(y_2-y.\text{predict}_2)^2+\ldots+(y_n-y.\text{predict}_n)^2}{n}}$$

这里以 6.2.1 节引例中的 110 个测试集数据为例，如表 6-17 所示。

表 6-17 训练集和测试集的模型效果

	训练集 data.train	测试集 data.test
预测	`train.pred<-predict(sol.rpart)`	`test.pred<-predict(sol.rpart, data.test)`
sol.error	`sqrt(mean((train.pred-data.train$y)^2))` **`[1] 3.014419`**	`sqrt(mean((test.pred-data.test$y)^2))` **`[1] 12.24977`**

6.4.3 过度拟合和剪枝

图 6-15 把 220 个样本点划分到 10 个不同区域中，理论上可以划分 220 个不同区域。然而虽然划分过细，训练集预测效果会好，但会使测试集（输入新数据样本）的预测效果降低，造成过分拟合错误。

对于一个过分拟合的模型，其测试集误差（代价）和回归树的规模（复杂）将会呈反

比。而一个合理的模型，其测试集误差（代价）和回归树的规模（复杂）都要尽可能地小。为了解决过度拟合问题，这里引入回归树 T 的代价复杂度概念，记为 $R_{cp}(T)$，其定义为：$R_{cp}(T)=R(T)+cp*|T_{num}|$。R(T) 是回归树 T 在测试集上的预测误差。$|T_{num}|$ 表示叶节点的数目。cp 是复杂度系数，表示每增加一个叶节点所带来的代价复杂度。每个决策点均可以计算出一个对应的复杂度系数 cp，并从叶节点至根节点不断变大。使用 sol.rpart$cptable 可以读取每次分裂后的 cp 的综合详情。代码如下：

```
sol.rpart$cptable
          CP   nsplit  rel error     xerror       xstd
1 0.48331629       0 1.00000000  1.0029004 0.09787582
2 0.19313964       1 0.51668371  0.5216048 0.04471342
3 0.10547606       2 0.32354407  0.3308626 0.03365875
4 0.03622948       3 0.21806801  0.2303217 0.03514681
5 0.03264176       4 0.18183853  0.1852319 0.03509433
6 0.01948026       5 0.19919677  0.1900906 0.03506241
7 0.01324142       7 0.21023625  0.2223089 0.02119021
8 0.01000000       9 0.28375342  0.2295836 0.02140874
```

nsplit 是分裂次数，由于 sol.rpart 模型中有 9 个决策节点，所以 nsplit 取 [0,9]。xerror 是通过交叉验证获得的模型误差，xstd 是模型误差的标准差。

剪枝实际就是找到一个合理的 cp 值，达到平衡测试集误差和树规模的目的，从而使其数值均很小。一般说来，随着拆分的增多（nsplit 增加），复杂性参量（cp）会单调下降（纯度越来越高），但是预测误差（xerror）则会先降后升，一般允许误差在一个标准差（xstd）内波动。这就产生了在剪枝理论中比较著名的规则：1- 标准差规则。其意思是：首先要保证预测误差尽量小，但不一定要取最小值，而是允许它在“最小的误差 ± 一个相应标准差”的范围内。然后在此范围内选取尽量小的复杂性参量，进而以它为依据进行剪枝。例如，由 sol.rpart$cptable 得到，在分裂 4 次 (nsplit=4) 后，模型误差（xerror）最小等于 0.1852319，其标准差（xstd）为 0.03509433。

```
          CP    nsplit  rel error     xerror       xstd
5 0.03264176       4 0.18183853  0.1852319 0.03509433
```

可以选取 xerror 在 0.1852319 ± 0.03509433 范围（即 [0.1501376, 0.2203262]）内 cp 的最小取值作为最合适的复杂度系数 cp。本例中处于该范围的是 nsplit=4，5 的情况：

```
          CP    nsplit  rel error     xerror       xstd
5 0.03264176       4 0.18183853  0.1852319 0.03509433
6 0.01948026       5 0.19919677  0.1900906 0.03506241
```

此时 nsplit=5 时的 cp 为最小值 0.01948026，所以希望剪除 nsplit=7 和 9 时的树，保留 nsplit=5 以上的树。

使用 prune(sol.rpart, 0.020) 函数进行剪枝，把 cp<0.020 的树剪除，最后得到：

```
prune(sol.rpart,0.020)$cptable
          CP   nsplit  rel error     xerror       xstd
1 0.48331629       0 1.00000000  1.0029004 0.09787582
```

```
2 0.19313964      1     0.51668371   0.5216048   0.04471342
3 0.10547606      2     0.32354407   0.3308626   0.03365875
4 0.03622948      3     0.21806801   0.2303217   0.03514681
5 0.03264176      4     0.18183853   0.1852319   0.03509433
6 0.02000000      5     0.19919677   0.1900906   0.03506241
```

注意，这里为了保留 nsplit=5 以上的树（cpmin=0.01948026），取 prune 函数的参数 cp 为 0.020，它略大于 0.01948026 且小于 0.03264176。

使用 plotcp 函数可以绘制出 cp 的波动关系，从而直观观察出合适的 cp 取值。

```
plotcp(sol.rpart, minline = TRUE, lty = 3, col = 1,upper = c("size", "splits",
"none"))
```

绘制结果如图 6-16 所示。

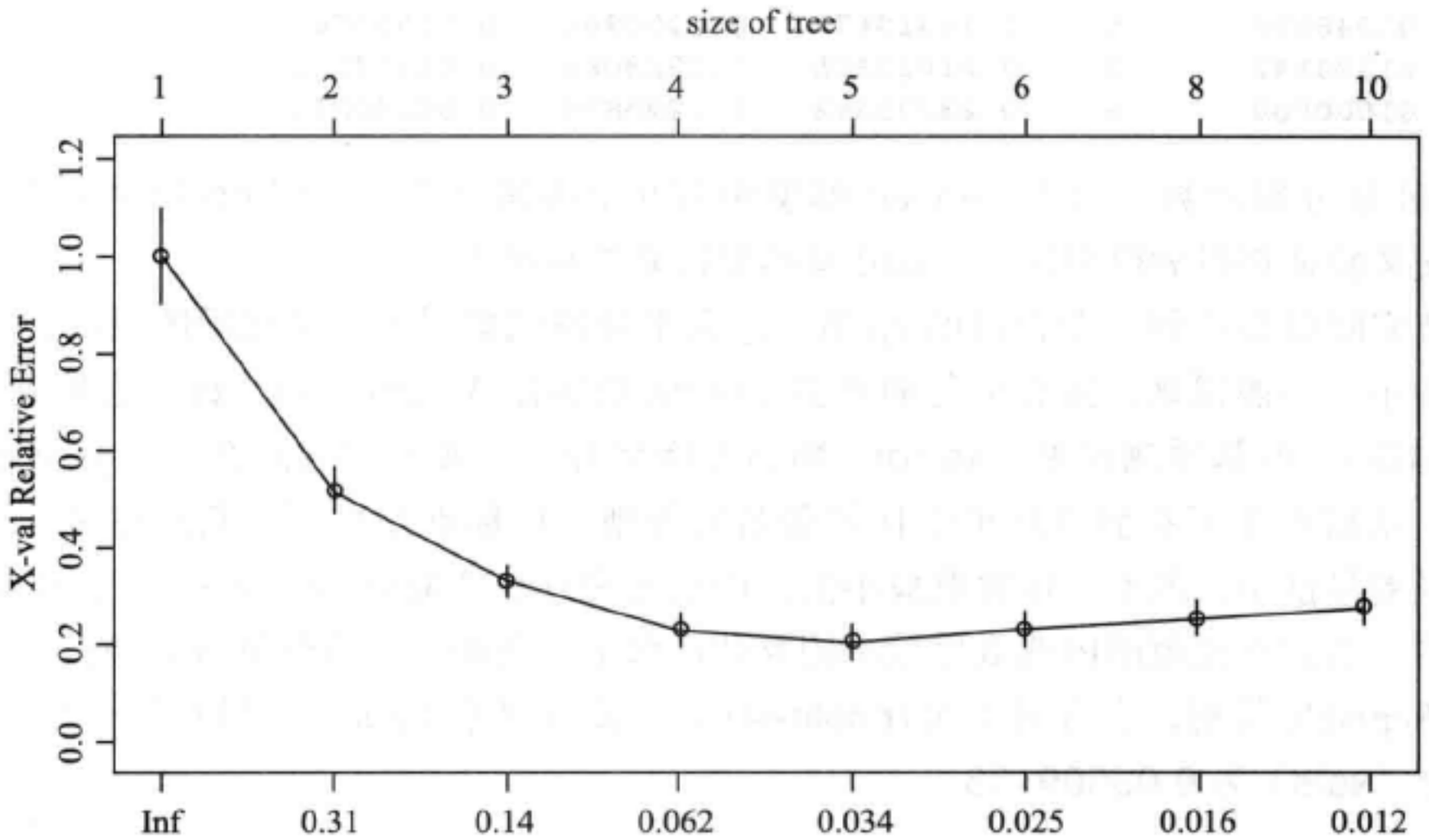


图 6-16　cp 波动图

其中 size of tree 是每次分裂都会产生相应的叶节点数目，由于 sol.rpart 中有 10 个叶节点，所以 size of tree 的取值范围是 [1,10]。例如，在分裂 9 次后（nsplit=9），对应的叶节点数为 10(size of tree=10）。sol.rpart 模型中不同 nsplit 对应的叶节点数（size of tree）如表 6-18 所示。

表 6-18　nsplit 和 size of tree 对照表

nsplit	0	1	2	3	4	5	7	9
size of tree	1	2	3	4	5	6	8	10

左侧 x-val relative error 是 sol.rpart$cptable 中的 rel error，其变化趋势与模型误差 sol.rpart$cptable 中的 xerror 相同。

第7章

分类指标建模：分类分析

上一章讲解了连续型指标的建模，本章详细讨论如何对一个分类指标建模，以及如何检验其模型的性能。本章讲解的大部分算法均可以用于计算目标变量被预测为正例和反例（以二元分类为例）的概率，从这个角度来看，这些分类器算法的本质也是回归算法，即大部分分类器也可以用于回归分析。

7.1 决策树分类分析

7.1.1 概述

决策树算法的输出通常类似于一套由 if-elase（二分支）或 case（多分枝）语句组成的判定式，因其画出的形状很像一棵树，故把此类算法统称为决策树算法，常用的决策树算法有 C4.5、CART 等。

1. 决策树

图 7-1 是可用于判断登录行为是否为恶意攻击的一组决策树（此图并非是真实的决策树结果，是为了方便说明而虚构的）。图中共有 17 个节点（node），它们分为两类节点结构：决策节点（non-leaf node）和叶节点（leaf node）。

决策节点用于指定要在哪个自变量上做检验。在图 7-1 中有 7 个决策节点，涉及 4 个自变量。通过不同自变量的检验及相应的取值，可以最终唯一地对应下方的叶节点。其中，决策节点 1 由于只有输出分支而没有任何输入分支，所以也称为根决策节点。

叶节点用于表示目标变量的最后类取值。在图 7-1 中，目标变量的类取值为：真实登录（类 1）、手动攻击（类 2），以及机器攻击（类 3）。

图 7-1 决策树对应的伪代码如下：

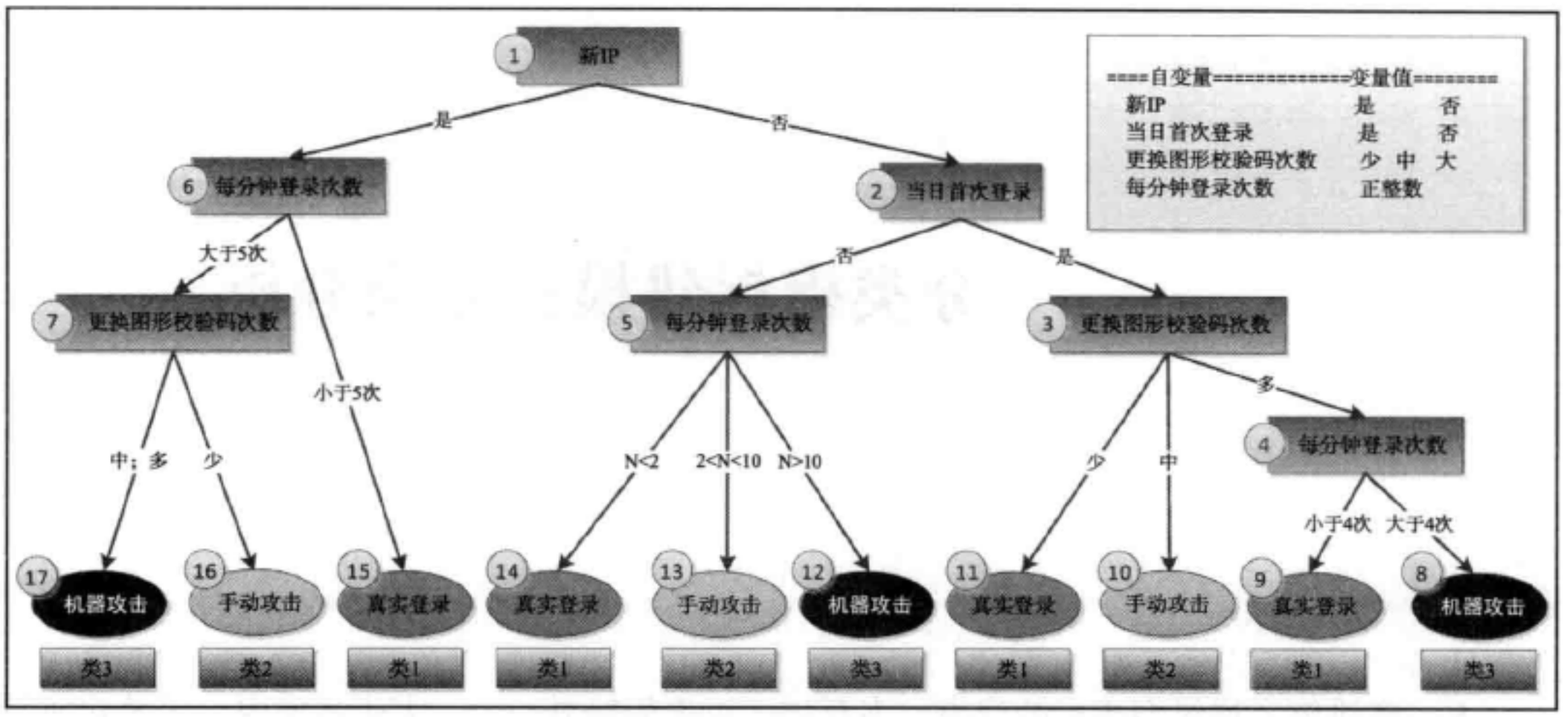


图 7-1 登录分类树示例图

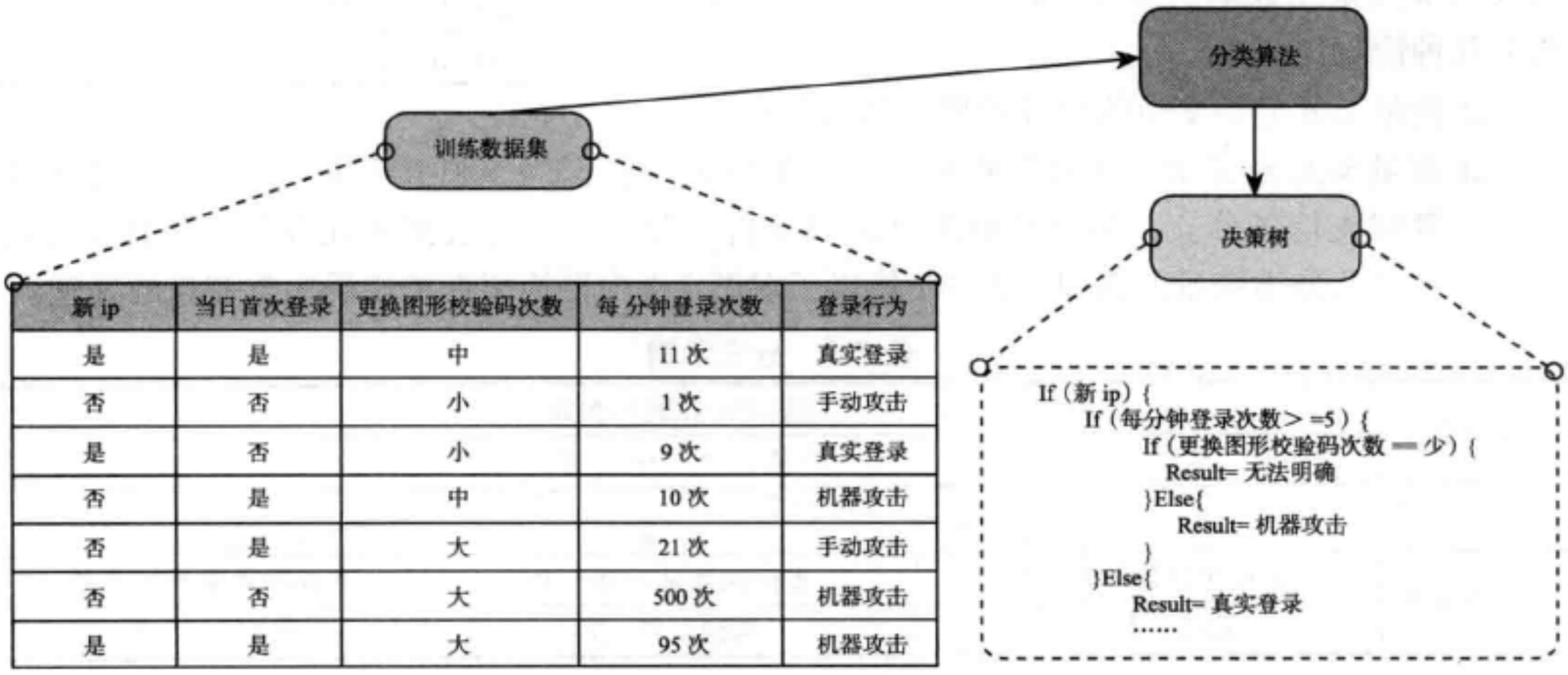
```
If(新 ip){------------------------------------------------------------决策节点 1
    If(每分钟登录次数 >=5){-----------------------------------------决策节点 6
        If(更换图形校验码次数 == 少 ){-----------------------------决策节点 7
            Result= 手动攻击 ------------------------------------叶节点 16
        }Else{
            Result= 机器攻击 ------------------------------------叶节点 17
        }
    }Else{
        Result= 真实登录 ------------------------------------------叶节点 15
    }
}Else{
    If(当日首次登录){ ----------------------------------------------决策节点 2
        Case(更换图形校验码次数){ ---------------------------------决策节点 3
            少: Result= 真实登录 ------------------------------------叶节点 11
            中: Result= 手动攻击 ------------------------------------叶节点 10
            多: If(每分钟登录次数 >=4){ ------------------------------决策节点 4
                    Result= 机器攻击 ------------------------------叶节点 8
                }Else{
                    Result= 真实登录 ------------------------------叶节点 9
                }
        }
    }Else{
        Case(每分钟登录次数){ ------------------------------------决策节点 5
            <=2:Result= 真实登录 ------------------------------叶节点 14
          >2 且 <=10: Result= 手动攻击 ------------------------------叶节点 13
            >10:Result= 机器攻击 ------------------------------叶节点 12
    }
}
```

程序员可以依据上面的伪代码，快速写出分类器函数，也可以拆分决策树，以得到一条

一条的判定规则集合。

2. 分类器模型建立

与回归分析一样，分类器模型的建立需要一组训练集数据（training dataset），并且要依据固定的算法和参数来建立，如图 7-2 所示。



新 ip	当日首次登录	更换图形校验码次数	每 分钟登录次数	登录行为
是	是	中	11 次	真实登录
否	否	小	1 次	手动攻击
是	否	小	9 次	真实登录
否	是	中	10 次	机器攻击
否	是	大	21 次	手动攻击
否	否	大	500 次	机器攻击
是	是	大	95 次	机器攻击

图 7-2 决策树建立过程

决策树的建立过程实际就是节点的分裂（split）过程，如下是建立决策树分类器模型归纳算法的伪代码：

```
BuildTree(S,A){
(S 是训练集样本数据，A 是自变量集合)
    If(stop_ split(S,A)==TRUE){---------------------- 判定为叶节点
        Leaf_node=creat_enode()--------------------- 建立节点
        Leaf_node .lable=Classify(S)---------------- 设定节点为叶节点并确定目标变量取值
    }else{------------------------------------------ 判定为决策节点
        A_split= Best_split(S,A)-------------------- 寻找最佳的分裂自变量 A_split
        Non_leaf_node=creat_enode()----------------- 建立节点
        For(v in V){
         (V 是自变量 A_split 的所有取值集合，v 是遍历 V 时的某种取值)
           Child_tree= BuildTree(Sv,A)------------- 以归纳算法建立子决策树 Child_tree
            (Sv 是 A_split 在 v 输出时的训练集样本)
        }
    }
    Return tree
}
```

其中 stop_ split 函数用于判定是否停止分裂。Create_node 函数用于建立节点。Classify 函数用于把节点确定为叶节点，并确定目标变量分类取值。Best_split 函数遍历所有自变量并寻找出最佳的分裂自变量 A_split。其中的关键步骤在于 Best_split 和 stop_ split 函数。

（1）算法如何分裂，以确定决策节点和分支（Best_split 函数）

这是在解决树的成长问题，首先要遍历所有自变量的分裂情况，并使用一套客观的度量指标（Measures）来衡量分裂后的效果。表 7-1 是 7.1 节涉及的决策树算法中的度量指标。

表 7-1　常用决策树的度量指标

算法	度量指标
C4.5	Gain Ratio 信息增益比率
CART	GINI
CHAID	卡方值

最后选择一个最佳的自变量（并确定其分支情况）作为决策节点来完成本次分裂。以图 7-1 为例，其中会涉及如下几种情况：

- 衡量二元分类变量的分裂效果，如决策点 2。
- 衡量多元分类变量的分裂效果，如决策点 3。对于 C4.5 和 ID3 等算法，其建立的决策树支持多分支，但是 CART 算法只支持二分支。此时会把多元分类变量转化为若干二元分类变量，例如，表 7-2 给出了对图 7-1 中更换图形校验码次数变量的处理。

表 7-2　分支示例

多分支	更换图形校验码次数		
	取值：少　中　多		
二分支转换	更换次数是否等于少	更换次数是否等于中	更换次数是否等于多
	取值：是 否	取值：是 否	取值：是 否

- 连续型变量中分支情况的确定以及分裂效果的衡量。例如，为何决策节点 5 被分裂为小于 2、小于 10、大于 10 的这 3 个分支。对于 C4.5 等（ID3 算法不支持连续变量分裂）算法，连续型变量可以分裂为多个分支（见图 7-3），但是 CART 算法只支持分为 2 支，如图 7-4 所示。

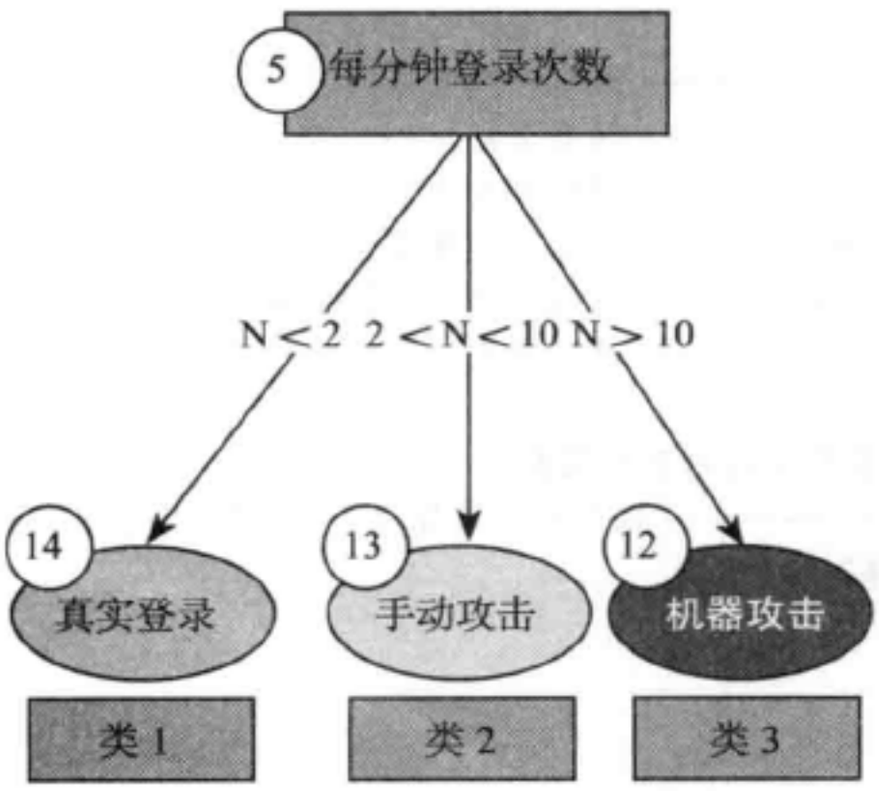


图 7-3　C4.5 多分支算法示例图

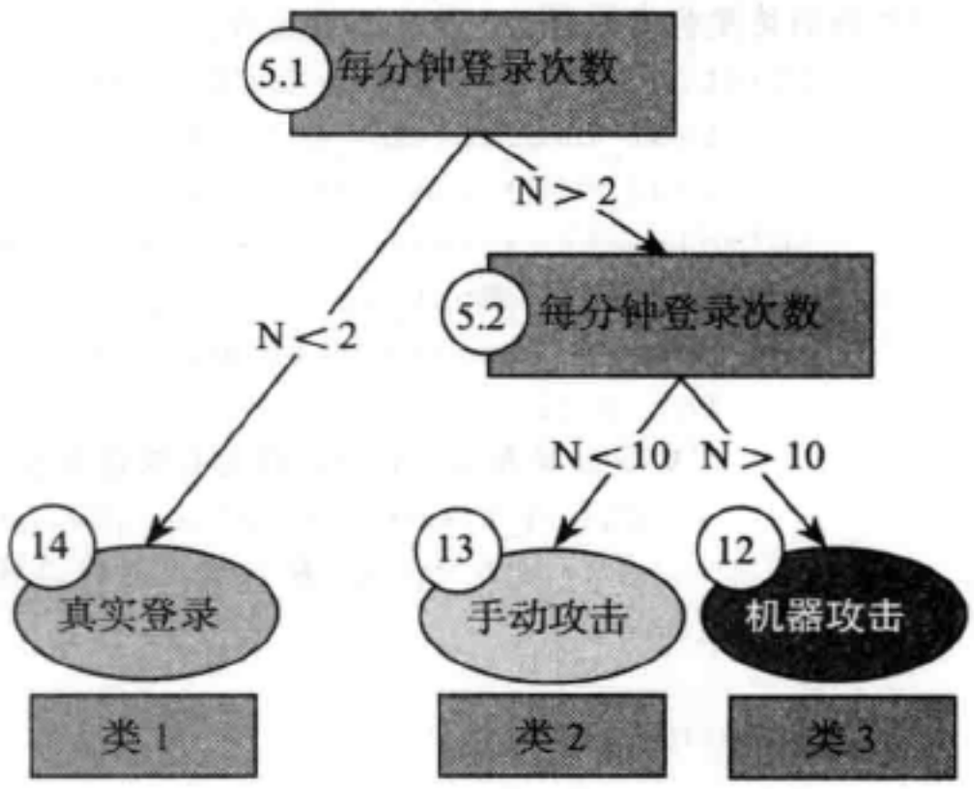


图 7-4　CART 二分支算法示例图

（2）何时停止分裂，以确定叶节点（stop_ split 函数）

决策树不能无限制地分裂生长下去，否则会出现严重的过分拟合现象。虽然在这种情况下，对训练集数据的分类预测功能很好，但是对于测试集的新数据，其预测性能很差。叶节

点中包含的样本数越少，显著性越差（由于偶然性而发生预测结果的可能性就越大），得到的结论也就越不可信。同时，叶节点内目标变量频数分布的不同，也会对结论的可信度产生巨大影响。例如，在对训练集做分类分析时，叶节点 8、叶节点 10 和叶节点 14 中的目标变量情况如表 7-3 所示。

表 7-3 预测目标变量示例

	目标变量样本数目				判定叶节点为
	真实登录	手动攻击	机器攻击	总和	
叶节点 8	3(2.4%)	20(16.4%)	99(81.2%)	122(100%)	机器攻击
叶节点 10	1(10%)	8(80%)	1(10%)	10(100%)	手动攻击
叶节点 14	41(42.3%)	36(37.1%)	20(20.6%)	97(100%)	真实登录

由表 7-3 可知，虽然节点 10 最终判定目标变量取值为“手动攻击”（因为其比例占 80%，比其他取值均高，这也是 Classify 函数的实现方式），但是其总体的样本数目较少（只有 10 个），所以其偶然发生的可能性很高，也就是说，结论的可信度并不高。同时虽然节点 14 最终判定目标变量取值为“真实登录”，但是其占比只有 42.3%，所以其可信度也不高。

Best_split 和 stop_split 函数将会在不同算法下详细讲解。

3. 模型剪枝

上面简要讲解了模型的建立方法，但是由于存在过分拟合等问题，所以往往要对模型进行剪枝。剪枝处理后，虽然训练集的错误率会有所增加，但是却能让新样本的预测错误率下降。剪枝又分为前剪枝和后剪枝。

- 前剪枝 preprune：在模型建立过程中就完成剪枝，即加强停止分裂的条件。
- 后剪枝 postprune：在模型建立之后才进行剪枝。在剪枝前要设置停止剪枝的相应参数，然后从模型的叶节点逐层向上剪枝。

4. 模型性能和预测

建立好决策树后，可以对新样本进行预测分类。例如一个 IP 的登录行为数据如表 7-4 所示。

表 7-4 IP 登录数据示例

新 IP	当日首次登录	更换图形校验码次数	每分钟登录次数	实际结果	预测结果
否	是	大	6 次	真实登录	机器攻击

根据图 7-1 来看，它符合决策树中 1 → 2 → 3 → 4 → 8 这个路径，最后预测的结果为机器攻击（叶节点 8）。但是其实际结果却是真实登录，所以预测错误了。衡量一个分类模型的性能不能单靠一个新样本的预测结果，要对大量新样本（称之为测试集 test dataset）进行预测，并就其错误率等数据建立一套衡量标准。

7.1.2 C4.5 算法

ID3 算法是分类分析中的鼻祖级算法，它以信息论中的信息增益为核心思想来建立分类

树模型。C4.5 算法实际上就是在 ID3 算法的基础上优化而成的。

1. 在 C4.5 算法中，如何分裂

信息熵（entropy）、信息增益（gain）和增益比率（gain ratio）均是从信息论中引申出来的概念。同时这 3 个概念也是 ID3 算法和 C4.5 算法的核心基石，其中增益比率是 C4.5 算法的分裂指标。

假设样本数据为 S，输入自变量为 Xn，目标变量为 Y。下面介绍这 3 个概念在如下示例中如何体现。

表 7-5 是 14 名用户的购买行为数据，用于预测是否购买。其中浏览时间为自变量 A_1，表示用户浏览该商品单品页的时间，单位为秒。网龄为自变量 A_2，表示用户使用互联网的时长，A 为小于 1 年，B 为 1 ～ 5 年，C 为 5 年以上。性别为自变量 A_3，1 为女性，0 为男性。是否购买为目标变量 Y，Yes 表示购买，No 表示不购买。

表 7-5 数据集

序列号	1	2	3	4	5	6	7	8	9	10	11	12	13	14
浏览时间 (A_1)	184	196	208	209	301	321	322	322	411	411	469	469	501	537
网龄 (A_2)	B	A	A	C	B	B	C	C	C	A	B	A	A	C
性别 (A_3)	1	1	0	1	0	0	0	0	1	0	1	1	0	0
是否购买 (Y)	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	No	No	Yes	No	No	Yes

（1）信息熵

先来看两个例子，如下：

1）分裂前，样本数据 S 的目标变量 Y 的信息熵为何值?

目标变量 Y 取值 yi 时的比例记为 $P\{Y=yi\}$，因为表 7-5 里 14 个样本中的目标变量 Y（用户是否购买）有 9 个取 Yes，即 $P\{Y=\text{Yes}\}=9/14$，5 个取 No，即 $P\{Y=\text{No}\}=5/14$，所以样本数据 S 的目标变量 Y 的信息熵定义为：

$$\text{Entropy}(S)=\sum P\{Y=yi\}\times\log2(1/P\{Y=yi\})=-\sum P\{Y=yi\}\times\log_2 P\{Y=yi\}$$

$$=-P\{Y=\text{Yes}\}\times\log_2 P\{Y=\text{Yes}\}-P\{Y=\text{No}\}\times\log_2 P\{Y=\text{No}\}$$

$$=-9/14\times\log_2(9/14)-5/14\times\log_2(5/14)=0.94$$

2）在自变量 A_i 下分裂后，样本数据 S 的目标变量 Y 信息熵为何值?

这里先把某一自变量 A_2 考虑进去，再来计算样本数据 S 的信息熵。表 7-6 为计算自变量 A_2 下样本数据 S 的目标变量 Y 的信息熵 $\text{Entropy}(S|A_2)$ 的过程。其中自变量 A_2 取 $a2j$ 值时的比例记为 $P\{A_2=a2j\}$，此时的样本数据记为 S_j,$\text{Entropy}(S_j)$，表示 A_2 取值 $a2j$ 时的目标变量 Y 的信息熵。

表 7-6 计算信息熵

序列号	2	3	10	12	13	1	5	6	11	4	7	8	9	14
网龄 A_2	A	A	A	A	A	B	B	B	B	C	C	C	C	C
是否购买 Y	Yes	Yes	No	No	No	Yes	Yes	Yes	Yes	No	Yes	Yes	No	Yes
$P\{A_2=a2j\}$	$P\{A_2=\text{A}\}=5/14$					$P\{A_2=\text{B}\}=4/14$				$P\{A_2=\text{C}\}=5/14$				

（续）

Entropy(S_j)	Entropy(S_A)=−2/5 × $\log_2$(2/5) −3/5 × $\log_2$(3/5)	Entropy(S_B)=−4/4 × $\log_2$(4/4) −0/4 × $\log_2$(0/4)	Entropy(S_C)=−3/5 × $\log_2$(3/5) −2/5 × $\log_2$(2/5)

Entropy($S|A_2$)= $\sum P(a2j) \times$ Entropy(S_j)

=$P\{A_2$=A$\} \times$ Entropy(S_A)+ $P\{A_2$=B$\} \times$ Entropy(S_B)+ $P\{A_2$=C$\} \times$ Entropy(S_C)

=5/14 × (−2/5 × $\log_2$(2/5)−3/5 × $\log_2$(3/5))+ 4/14 × (−4/4 × $\log_2$(4/4)−0/4 × $\log_2$(0/4))+ 5/14 × (−3/5 × $\log_2$(3/5)− 2/5 × $\log_2$(2/5))

=0.69

看了两个例子，现在确定信息熵的含义。假设变量 Y 可以取 0 和 1 两个值，在不同的频数分布时其信息熵为：

变量 Y	计数
取值 =0	0
取值 =1	10

Entropy=−0/10* $\log_2$(0/10) −10/10* $\log_2$(10/10)=0

变量 Y	计数
取值 =0	5
取值 =1	5

Entropy=−5/10* $\log_2$(5/10) −5/10* $\log_2$(5/10)=1

变量 Y	计数
取值 =0	10
取值 =1	0

Entropy=-10/10* $\log_2$(10/10) −0/10* $\log_2$(0/10)=0

可见变量 Y 的不纯度越低（即纯度高，Y 变量全部取 1 或 0）时，信息熵越小，最小值为 0。变量 Y 的不纯度越高（即纯度低，Y 变量一半取 1，一半取 0），信息熵越大，最大值为 1。因此，信息熵可用于衡量变量的不纯度。

（2）信息增益

在使用决策树算法选择最佳自变量 A_i 进行分裂时，我们认为分裂后目标变量 Y 的纯度越高越好，即不纯度越低越好。此时可引入信息增益（gain）的概念，用于衡量自变量 A_i 分裂后的效果。

Gain(A_i)= 分裂前目标变量 Y 的信息熵 − 对自变量 A_i 分裂后目标变量 Y 的信息熵

= Entropy(S) − Entropy($S|A_i$)

Gain(A_i) 正比于自变量 A_i 分裂后目标变量 Y 的纯度。在 ID3 算法中，Best_split 函数会遍历每个自变量分裂后的信息增益，并从中挑选出信息增益最大的自变量作为分裂节点。在表 7-5 中，自变量 A_2 分裂后的信息增益为：

$$\text{Gain}(A_2)= \text{Entropy}(S) - \text{Entropy}(S|A_2)=0.94-0.69=0.25$$

使用信息增益作为分裂指标的缺点是：它偏向于选择取值较多的自变量作为分裂节点，针对这一问题，C4.5 算法引入增益比率作为分裂指标来协助解决。

（3）增益比率

为了消除 A_i 取值数目对信息增益 Gain(A_i) 的影响，C4.5 算法引入了增益比率（gain ratio）这一指标，即在原有信息增益的基础上除以包含取值数目信息的 splitINFO。SplitINFO(A_i) 是样本数据 S 中自变量 A_i 的信息熵，表 7-5 中自变量 A_2 的 splitINFO (A_2) 的值如表 7-7 所示。

表 7-7　计算 splitINFO

序列号	1	2	3	4	5	6	7	8	9	10	11	12	13	14
网龄 (A_2)	B	A	A	C	B	B	C	C	C	A	B	A	A	C
A_2 的信息熵	$=-P\{A_2=A\}*\log_2 P\{A_2=A\}-P\{A_2=B\}*\log_2 P\{A_2=B\}-P\{A_2=C\}*\log_2 P\{A_2=C\}$													
splitINFO (A_2)	$=-5/14*\log_2(5/14)-4/14*\log_2(4/14)-5/14*\log_2(5/14)=1.57$													

增益比率的定义为：

$$\text{gain ratio}(A_i)=\text{Gain}(A_i)/\text{SplitINFO}(A_i)$$

在本节示例中分裂自变量 A_2 后，目标变量 Y 的信息增益比率为：

$$\text{gain ratio}(A_2)=\text{gain}(A_2)/\text{splitINFO}(A_2)=0.25/1.57=0.16$$

（4）连续变量处理

对于连续型自变量的分裂，C4.5 算法首先要把这些连续变量的取值分为不同的区间，即把连续类型离散化。离散化的大致方法如下：

1）把连续型自变量的最小值和最大值分别赋值给 Min 和 Max。

2）设置区间 [Min，Max] 中的 N 个等分断点 Xi，即：

$$Xi=\min+i*(\max-\min)/N, \text{其中 } i=1，2，3,\cdots,N$$

3）分别计算把 [Max, Xi] 和 (Xi, Max] 作为区间值时的信息增益。

4）选取信息增益取值最大的 Xk 作为该连续型自变量的断点，并把分支设为 [Max, Xk] 和 (Xk, Max] 两个区间值。

（5）C4.5 的 Best_split 函数

有了增益比率的基础，理解 C4.5 算法的 Best_split（S,A）函数就很容易了，它用于分裂出最佳的决策节点 A_split。该函数的实现过程可以概括为：遍历所有自变量 A，计算各变量 A_i 分裂后目标变量 Y 的增益比率 gain ratio(A_i)，并选取其值最大的自变量作为最佳的决策节点 A_split。

2. 在 C4.5 算法中，如何停止分裂

C4.5 算法中的 stop_ split 函数用于停止分裂，实现该函数的主要条件有：

- 一个节点的所有目标变量样本值均为同一类别。
- 若无自变量可用于划分当前样本集，则将当前节点强制划为叶节点，并把样本中数目最多的类别标记为预测类别。
- 如果分裂后剩余的样本数小于某个给定阈值 Minimun_Support，则返回一个叶节点，并把众数类别标记为预测类别。例如，取值为 10，即如果剩余样本数小于 10，则自动将其归为叶节点，停止分裂。这实际上就是一种前剪枝的实施方式。

3. 后剪枝算法

C4.5 算法支持置信度和减少 – 误差这两种后剪枝算法。

（1）置信度剪枝

置信度（confidence）可以通俗地理解为可信度（取值为 0 ～ 1，典型值为 0.75），在决

策树中的每个节点均可以计算出其置信度。越是接近底层（叶节点），其置信度越小，即结论越不可信。反之，越是接近顶层（根节点），其置信度越大，即结论越可信。在进行置信度剪枝前，先设置最小置信度阈值 min_ confidence，然后从底部依次比较各节点的置信度，小于 min_ confidence 的节点则进行剪枝，大于 min_ confidence 的节点则停止剪枝。可见 min_ confidence 设置得越大（接近 1），剪枝强度越大，树越小。反之，min_ confidence 越小（接近 0），剪枝强度越小，树越大。

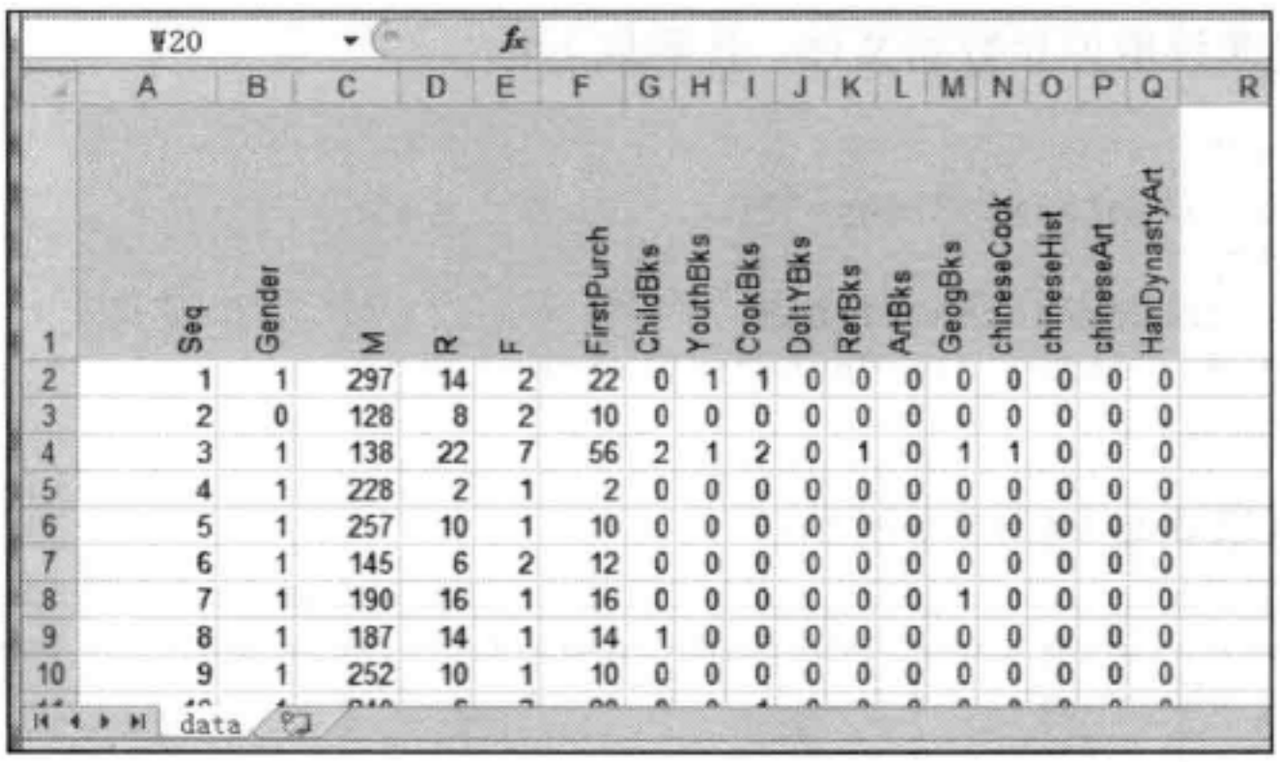
（2）减少－误差剪枝

通过测试集数据，可以计算决策树中每个节点的误差 e_i。减少－误差（Reduce-Error）剪枝算法就是使用测试集数据，先计算待剪子树的父决策节点的误差 e_i，然后计算待剪子树内所有叶节点的加权误差之和，其定义为 $\sum p_i * e_i$。其中 p_i 是第 i 个子叶节点样本数占整个子树内样本数的比例，e_i 是子叶节点的误差。最后将 $\sum p_i * e_i$ 与父决策节点的误差进行比较，如果 $\sum p_i * e_i$ 大于父决策节点的误差，则可以对这些子树的叶节点剪枝，否则不能剪除。在实际操作中会把训练集数据折成 N（默认为 3）份，取其 1 份作为计算误差 e_i 的测试集数据，另外几份用于建立决策树模型。

4. 数据集

（1）数据集简介

为了更好地理解后续分类器的概念，首先引入一个数据集示例（在后续章节中会多次使用）。该数据集是某购书类网站针对新书《西汉艺术简史》的 4000 个调研数据，格式如图 7-5 所示（使用 Excel 打开的部分截图）。



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Seq	Gender	M	R	F	FirstPurch	ChildBks	YouthBks	CookBks	DoItYBks	RefBks	ArtBks	GeogBks	chineseCook	chineseHist	chineseArt	HanDynastyArt	
2	1	1	297	14	2	22	0	1	1	0	0	0	0	0	0	0	0	
3	2	0	128	8	2	10	0	0	0	0	0	0	0	0	0	0	0	
4	3	1	138	22	7	56	2	1	2	0	1	0	1	1	0	0	0	
5	4	1	228	2	1	2	0	0	0	0	0	0	0	0	0	0	0	
6	5	1	257	10	1	10	0	0	0	0	0	0	0	0	0	0	0	
7	6	1	145	6	2	12	0	0	0	0	0	0	0	0	0	0	0	
8	7	1	190	16	1	16	0	0	0	0	0	0	1	0	0	0	0	
9	8	1	187	14	1	14	1	0	0	0	0	0	0	0	0	0	0	
10	9	1	252	10	1	10	0	0	0	0	0	0	0	0	0	0	0	

图 7-5　数据集截图

其中各列的含义如表 7-8 所示。

表 7-8　数据集格式

列名称	含义	格式
Seq	序号，对于数据分析没有意义	
Gender	性别	分类型数据：0= 男，1= 女

（续）

列名称	含义	格式
M	消费信息：花在书上总金额	连续型数据
R	最近消费信息：距离最后一次购买的月数	连续型数据
F	消费频次——总购买次数	连续型数据
FirstPurch	第一次购买至今的月数	连续型数据
ChildBks	购买儿童类图书的数量	连续型数据
YouthBks	购买青年类图书的数量	连续型数据
CookBks	购买厨艺类书的数量	连续型数据
DoItYBks	购买自助类书的数量	连续型数据
RefBks	购买参考类书（地图、百科辞典、字典）的数量	连续型数据
ArtBks	购买艺术类书的数量	连续型数据
GeogBks	购买地理类书的数量	连续型数据
chineseCook	《中国菜系大观》一书的购买数量	连续型数据
chineseHist	《中国历史概要》一书的购买数量	连续型数据
chineseArt	《中国艺术史》一书的购买数量	连续型数据
HanDynastyArt	是否购买《西汉艺术简史》一书	分类型数据：0= 未购买，1= 购买

这里以分类变量 HanDynastyArt 为目标变量，下面建立一个分类器模型，以便更好地预测用户是否购买新书《西汉艺术简史》这一行为。

1）源数据文件的读入。

原始数据存储在路径“D: \cbc”（Windows 系统）下 data.xls 文件中的 data 工作表中。这里使用 RODBC 包来读取原始数据文件，代码如下：

```
root<-"D:/cbc/"
file<-paste(root,"data.xls",sep="")
library (RODBC)
excel_file <- odbcConnectExcel(file)
data<- sqlFetch ( excel_file,"data")[,1:17]
close ( excel_file )
```

得到数据框对象 data 如图 7-6 所示。

```
> data[1:10,]
   Seq Gender   M  R F FirstPurch ChildBks YouthBks CookBks DoItYBks RefBks ArtBks GeogBks chineseCook chineseHist chineseArt HanDynastyArt
1    1      1 297 14 2         22        0        1       1        0       0       0       0           0           0          0             0
2    2      0 128  8 2         10        0        0       0        0       0       0       0           0           0          0             0
3    3      1 138 22 7         56        2        1       2        0       1       0       1           1           0          0             0
4    4      1 228  2 1          2        0        0       0        0       0       0       0           0           0          0             0
5    5      1 257 10 1         10        0        0       0        0       0       0       0           0           0          0             0
6    6      1 145  6 2         12        0        0       0        0       0       0       0           0           0          0             0
7    7      1 190 16 1         16        0        0       0        0       0       0       1           0           0          0             0
8    8      1 187 14 1         14        1        0       0        0       0       0       0           0           0          0             0
9    9      1 252 10 1         10        0        0       0        0       0       0       0           0           0          0             0
10  10      1 240  6 3         20        0        0       1        0       0       0       0           0           0          0             0
```

图 7-6　结果展示图

2）部分数据处理。

由于性别（Gender）及是否购买新书（HanDynastyArt）均是分类变量，所以使用 as.factor

函数将其转化为分类变量。代码如下：

```
data$Gender<-as.factor(data$Gender)
data$HanDynastyArt<-as.factor(data$HanDynastyArt)
```

由于购买《中国菜系大观》、《中国历史概要》、《中国艺术史》这3本书的用户数据chineseCook、chineseHist以及chineseArt本身是连续型数据，为了简化问题，这里把它们转化为分类型数据：如果等于0，则取值为0，表示未购买上述的图书，否则取值为1，表示已经购买。代码如下：

```
data$chineseCook<-as.factor(ifelse(data$chineseCook==0,0,1))
data$chineseHist<-as.factor(ifelse(data$chineseHist==0,0,1))
data$chineseArt<-as.factor(ifelse(data$chineseArt==0,0,1))
```

3）目标分类不均衡的抽样处理。

首先研究目标变量各分类取值的数目：

```
table(data$HanDynastyArt)
   0    1
3662  338
```

可见，在4000个样本数据中，3662个样本的目标变量取0（占整体的91.55%），338个取1（占整体的8.45%），其比例很不均衡。根据此类训练集样本数据建立的分类器会倾向于预测数目较多的一类。即，data$HanDynastyArt取0的情况，而实际上我们更加关心data$HanDynastyArt取1的情况。此时可以在data$HanDynastyArt取1的样本中重复抽取样本，使data$HanDynastyArt取0和1的数目相等。代码如下：

```
balance<-function(data,yval){
    y.vector<-with(data,get(yval))
    #在data数据框中读取列名称为yval参数的向量。例如，如果yval设置为HanDynastyArt，则
y.vector就是data$HanDynastyArt
    index.1<-which(y.vector==1)
    index.0<-which(y.vector==0)
    index.1<-sample(index.1,length(index.0),replace=T)
    result<-data[sample(c(index.0,index.1)),]
    result
}
data<-balance(data,"HanDynastyArt")
```

这里的自定义函数名称为balance，参数data是数据框对象，yval是data中目标变量的列名称。y.vector<-with(data,get(yval))表示在data数据框中读取列名称为yval的向量。例如，这里data应该取data，yval应该取"HanDynastyArt"，则y.vector就是data$HanDynastyArt。然后读取目标变量为1和0的序列号index.1和index.0，并对index.1重复抽样，抽样数目为index.0的数目，以使输出的数据1和0样本平衡。最后使用result<-data[sample(c(index.0,index.1)),]输出抽样数据。

进行转换后，目标变量各分类取值的数目为：

```
table(data$HanDynastyArt)
   0    1
3662 3662
```

4）训练集和测试集划分

一般情况下，使用训练集来建立模型，使用测试集来检验模型的性能，其划分比例为 1:1 或 7:3。这里自定义一个函数 apart.data，用于划分训练集和测试集数据。参数 data 是划分前的数据框，train.data.persent 是训练集占 data 的比例，默认为 0.7（即训练集：测试集=0.7:0.3=7:3）。代码如下：

```
apart.data<-function(data,train.data.persent=0.7){
    train.index<-sample(c(1:nrow(data)),round(nrow(data)*train.data.persent))
    data.train<-data[train.index,]
    data.test<-data[-c(train.index),]
    result<-list(train=data.train,test=data.test)
    result
}
p.data<-apart.data(data)
data.train<-p.data$train
data.test<-p.data$test
```

最后把原有数据框 data 分割为训练集数据框 data.train 和测试集数据框 data.test，样本数的比例约为 7:3。其中训练集目标变量 0 和 1 的数目分别为 2577 和 2550，合计样本数为 5127，如下所示：

```
table(data.train$HanDynastyArt)
   0    1
2577 2550
```

5. 在 R 语言中实现 C4.5 算法

C4.5 算法在另一个数据挖掘开源软件 weka 已经有很成熟的实现，在 R 语言中，可以通过 RWeka 包中的 J48 函数来调用 weka 的 C4.5 算法。

J48() 函数的基本形式为：J48(formula, data, subset, na.action,control = Weka_control())，参数说明如表 7-9 所示。

表 7-9　J48 函数参数说明

参数	说明
formula	分类方程形式，例如 $Y \sim X1+X2+X3$，表示使用自变量 $X1$、$X2$、$X3$（连续型或分类型）来对目标变量 Y（只能是分类型，即因子 factor 类型对象）做决策树模型
data	包含 formula 参数中变量的数据框
subset	用于在 data 中选取部分样本（部分行数据）
na.action	缺失数据的处理方法，默认是删除目标变量的数据，保留自变量的数据

（续）

参数	说明	
control	C4.5 算法的参数，使用 Weka_control() 函数进行设置。使用代码 WOW(J48) 来查看内部的参数列表，具体如下：	
	M	minNumObj ： 为停止分裂条件中叶节点数目的最小阈值 Minimun_Support（参见 7.1.2 节中的停止分裂条件 3 ）。M 值越大，树越小；反之，树越大，默认为 2
	B	binarySplits：设置是否进行二分支分裂。默认取值为 FALSE，即进行多分支分裂
	U	unpruned：设置修剪是否需要。默认为 FALSE，即做剪枝
	R	reducedErrorPruning：设置是否使用减少 - 误差（Reduce-Error）剪枝算法。J48 支持两种剪枝方法：置信度剪枝（R 设置为 TRUE）和减少 - 误差剪枝（R 设置为 FALSE）算法
	C	Confidence Factor ： 是用于置信度修剪法的置信因子，默认值为 0.25。1-C 就是参数 min_ confidence（参见 7.1.2 节中的置信度剪枝部分）。由于 1-C=min_ confidence，所以 C 越大，剪枝强度越小，即树越大；C 越小，树越小 注意：只有在使用置信度剪枝算法时（R 设置为 FALSE），C 才可以被设置，否则 C 不能设置数值
	N	在使用减少 - 误差剪枝算法时，系统把训练集拆为 N 份，其中一份用于计算各节点的误差，另外的数据用于建立决策树模型（参见 7.1.2 节中的减少 - 误差剪枝部分），默认值为 3 注意：只有在使用减少 - 误差剪枝算法时（R 设置为 TRUE），N 才可以被设置，否则 N 不能设置数值
	Q	seed , 是执行减少 - 误差剪枝算法时，用于计算节点误差数据的随机化种子，默认值为 1 注意：只有在使用减少 - 误差剪枝算法时（R 设置为 TRUE），Q 才可以被设置，否则 Q 不能设置数值
	S	SubtreeRaising：设置修剪树时是否考虑子树上升操作，默认设置为 True
	L	useLaplace：设置叶节点是否基于拉普拉斯平滑，默认设置为 False
	J	MDL 剪枝法：这是 J48 函数扩展的剪枝方法。它利用最短描述长度（Minimum Description Length）原则来对决策树进行剪枝。其核心思想是：最简单的就是最好的。优点是：在剪枝时无需额外的独立测试数据集

下面使用 7.1.2 节中的数据集来做 C4.5 算法的建模试验，注意目标变量 data$HanDynasty-Art 必须是 factor 因子类型。代码如下：

```
    mod.formula<-as.formula("HanDynastyArt ~ Gender+M+R+F+FirstPurch+ChildBks+YouthB
ks+CookBks+DoItYBks+RefBks+ArtBks+GeogBks+chineseCook+chineseHist+chineseArt")
    library("RWeka")
    c45.sol <-J48(mod.formula,data=data.train,control = Weka_control(U=T)); c45.sol
    J48 unpruned tree
    ------------------
    ArtBks <= 1
    |   M <= 437
    |   |   chineseArt = 0
    <中间部分省略展示>
    |   |   |   |   CookBks > 2
    |   |   |   |   |   R <= 10
    |   |   |   |   |   |   R <= 4: 0 (12.0/2.0)
    |   |   |   |   |   |   R > 4: 1 (20.0)
    |   |   |   |   |   R > 10: 0 (9.0)
    |   |   |   ArtBks > 2: 1 (72.0/1.0)
    Number of Leaves  :     483
    Size of the tree :      965
```

上面的代码输出了根据 C4.5 算法得到的决策树，J48 函数的 control 参数设置为 Weka_control(U=T)，即表示不使用任何剪枝算法。在代码的最后，“Number of Leaves: 483”表明叶节点数为 483，“Size of the tree: 965”表明共有 965 个节点。

ArtBks > 2: 1 (72.0/1.0) 表示在相应的叶节点下，训练集中有 72 个样本。模型预测该类用户会购买新书（预测目标变量 HanDynastyArt 取 1），其中 1 个样本的实际取值为 0（未购买），即预测错误了，另外 71 个样本则预测正确，如图 7-7 所示。

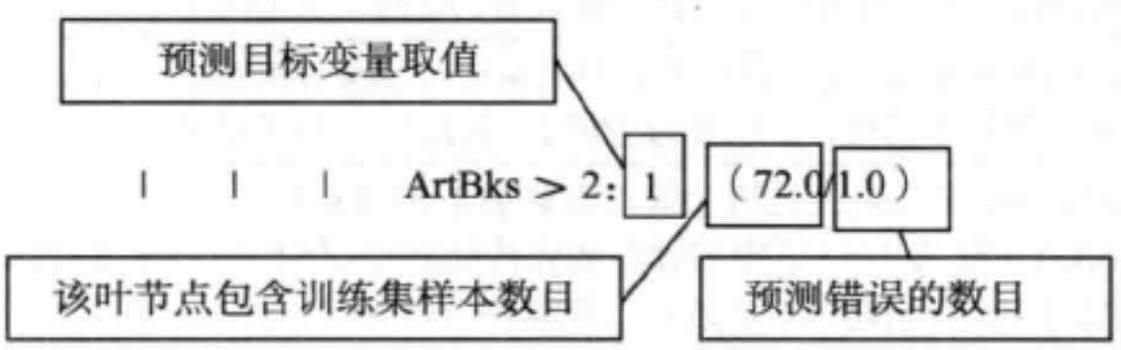


图 7-7　节点详述图

R <= 4: 0 (12.0/2.0) 表示在相应的叶节点下，训练集中有 12 个样本。模型预测该类用户不会购买新书（预测目标变量 HanDynastyArt 的取值为 0），其中 2 个样本的实际取值为 1（购买），即预测错误了，另 10 个样本预测正确。

使用 summary 函数查看模型概况。代码如下：

```
summary(c45.sol)
=== Summary ===
Correctly Classified Instances        5050               98.4981 %
Incorrectly Classified Instances        77                1.5019 %
                    <中间部分省略展示>
Total Number of Instances             5127
=== Confusion Matrix ===
    a    b   <-- classified as
 2500   77 |    a = 0
    0 2550 |    b = 1
```

使用 summary 函数可以查看到，在训练数据集上使用决策树的正确率（Correctly Classified Instances）为 98.4981 %，错误率（Incorrectly Classified Instances）为 1.5019 %。样本数（Total Number of Instances）为 7324。Confusion Matrix 是混淆矩阵，其中列是模型预测值，行是样本真实值，并且，在矩阵中正对角线的数值越大越好。从上面的混淆矩阵可知：

- 在训练集样本中，目标变量实际取值为 0 和 1 的数目依次是 2577 和 2550。
- 决策树预测训练集样本中有 2627 个取 1（购买新书），其中 2550 个预测正确了（实际也是 1），另有 77 个预测错误了（实际是 0）。
- 决策树预测训练集样本中有 2500 个的取值为 0（未购买新书），其中 2500 个预测正确了（实际也是 0），另有 0 个预测错误了（实际是 1）。

下面添加阈值参数的限制条件：

```
c45.sol<-J48(mod.formula,data=data.train,control = Weka_control(U=T,M=30));c45.sol
  <前面部分省略展示>
Number of Leaves  :     87
Size of the tree :     173
```

上面代码在停止分裂条件中设置了阈值参数 Minimun_Support=30（如果分裂后剩余的样本数小于 30，则返回一个叶节点），得到仅有 87（<483）个叶节点的决策树，即做了前剪枝。

```
c45.sol<-J48(mod.formula,data=data.train);c45.sol
<前面部分省略展示>
Number of Leaves  :     412
Size of the tree :     823
```

上面代码中未设置任何 control 参数，默认对其使用置信度为 0.75（置信因子是 0.25）的置信度剪枝算法，得到仅有 412（<483）个叶节点的决策树，即做了后剪枝。

```
    c45.sol<-J48(mod.formula,data=data.train,control = Weka_control(C=0.05));c45.
sol
    <前面部分省略展示>
    Number of Leaves  :     353
    Size of the tree :     705
```

上面代码把置信度剪枝算法的置信因子减小了（设为 0.05），即加大置信度为典型值 0.95。得到有 353（<412）个叶节点的决策树，即剪枝的强度比置信因子被设为 0.25 时要大很多（决策树的叶节点数目变少了）。

```
    c45.sol<-J48(mod.formula,data=data.train,control = Weka_control(R=T,N=4));c45.sol
    <前面部分省略展示>
    Number of Leaves  :     337
    Size of the tree :     673
```

上面代码使用了减少－误差剪枝算法，其中训练集中的 3/4 个样本用于建立决策树模型，1/4 个样本用于计算决策树内各个节点的误差（N=4），得到仅有 337（<483）个叶节点的决策树，即做了后剪枝。

也可以综合使用前剪枝和后剪枝。例如：

```
    c45.sol<-J48(mod.formula,data=data.train,control = Weka_control(C=0.1,M=30));c45.
sol
    <前面部分省略展示>
    Number of Leaves  :     71
    Size of the tree :     141
```

上面代码设置 control 参数为 Weka_control(C=0.1,M=30)，既使用了约束叶节点数目的前剪枝算法（Minimun_Support=30），也使用了置信因子为 0.1（最低置信度 min_confidence=0.9）的后剪枝算法。不过值得注意的是，两种后剪枝算法只能选择一个使用，将通过 R 参数来设置。使用 plot 函数可以绘制其树形结构图。

6. 预测

使用 J48 函数得到相应的 C4.5 算法模型后，可以使用 predict 函数来预测新样本的预测值。相关代码如下：

```
pre.c45<-predict(c45.sol)
```

上面代码对原始训练集 data 进行预测，返回的 pre.c45 是一个取值为 0 或 1 的 factor 分类向量。也可以使用新的测试集数据 test.data 进行预测。代码如下：

```
pre.c45<-predict(c45.sol,test.data)
```

不过测试集中需要以数据框为对象，其列名称也要和训练集统一。

注意

C4.5 算法的特点为：

输入变量（自变量）：为分类型变量或连续型变量。

输出变量（目标变量）：为分类型变量。

连续变量处理：N 等分离散化。

树分枝类型：多分枝。

分裂指标：信息增益比率 gain ratio（分裂后的目标变量取值变异较小，纯度高）。

前剪枝：叶节点数是否小于某一阈值。

后剪枝：使用置信度法和减少 - 误差法。

7.1.3　CART 算法

CART 的全称为 Classification And Regression Tree，即分类回归树。它既可以对分类型目标变量建模（分类分析），也可以对连续型目标变量建模（回归分析），下面介绍其分类分析的原理。

与 C4.5 算法类似，CART 算法也使用目标变量的纯度来分裂决策节点，只是它使用的分裂度量是 Gini 增益。需要注意的是，CART 内部只支持二分支树。

1. 在 CART 算法中，如何分裂

Gini 增益的本质也是对目标变量的纯度进行度量，仍使用表 7-5 中的数据来进行说明。

假设样本数据为 S，输入自变量为 Xn，目标变量为 Y。那么：

1）分裂前，样本数据 S 的目标变量 Y 的 Gini 增益为何值？

如果目标变量 Y 为 y_i 时的比例记为 $P\{Y=y_i\}$，则 Gini 的定义为：

$$\begin{aligned}\text{Gini}(S) &= 1-\sum P\{Y=y_i\}^2 \\ &= 1-P\{Y=\text{Yes}\}^2-P\{Y=\text{No}\}^2 \\ &= 1-(9/14)^2-(5/14)^2=0.459\end{aligned}$$

2）在自变量 A_i 下分裂后，样本数据 S 的目标变量 Y 的 Gini 增益为何值？

这里先把某一自变量 A_2 考虑进去，再计算样本数据 S 的 Gini。表 7-10 为计算自变量 A_2 下样本数据 S 的目标变量 Y 的增益 Gini($S|A_2$) 的过程。其中自变量 A_2 为 $a2j$ 时的比例记为 $P\{A_2=a2j\}$，此时的样本数据记为 S_j，Gini(S_j) 是指 A_2 为 $a2j$ 时的目标变量 Y 的 Gini。

表 7-10 计算 Gini

序列号	2	3	10	12	13	1	5	6	11	4	7	8	9	14
网龄 A_2	A	A	A	A	A	B	B	B	B	C	C	C	C	C
是否购买 Y	Yes	Yes	No	No	No	Yes	Yes	Yes	Yes	No	Yes	Yes	No	Yes
$P\{A_2=a2j\}$	$P\{A_2=A\}=5/14$					$P\{A_2=B\}=4/14$				$P\{A_2=C\}=5/14$				
Entropy(S_j)	$Gini(S_A)=1-(2/5)^2-(3/5)^2=0.48$					$Gini(S_B)=1-(4/4)^2-(0/4)^2=0$				$Gini(S_C)=1-(3/5)^2-(2/5)^2=0.48$				

$Gini(S|A_2)=\sum P(a2j)*Gini(S_j)$

$=P\{A_2=A\}*Gini(S_A)+P\{A_2=B\}*Gini(S_B)+P\{A_2=C\}*Gini(S_C)$

$=5/14*0.48+4/14*0+5/14*0.48$

$=0.343$

现在来看看 Gini 增益的意义。与 C4.5 中的信息熵类似，Gini 增益也是用于衡量目标变量的不纯度的。

CART 决策树算法在选择最佳自变量 A_i 进行分裂时，其分裂指标为 Gini 增益：

$\Delta Gini(A_i)$= 分裂前目标变量 Y 的 Gini- 对自变量 A_i 分裂后目标变量 Y 的 Gini

$=Gini(S)-Gini(S|A_i)$

仍以表 7−10 中的实例为例，对 A_2 变量进行分裂后，其 Gini 增益为

$$Gini(S)-Gini(S|A_2)=0.459-0.343=0.116$$

与 C4.5 算法使用的信息增益一样，Gini 增益作为分裂指标也具有“偏向于选择取值较多的自变量作为分裂节点”的缺点。针对这一问题，C4.5 算法引入增益比率（gain ratio）作为分裂指标来解决这个问题，而 CART 算法是通过限制只能进行二分支分裂来解决的（这也是 CART 算法只能做二分支决策树的主要原因）。

CART 算法对连续型变量的处理也与 C4.5 算法类似，只是离散化的指标不再是信息增益 Gain 而是 ΔGini（Gini 增益）。另外，CART 算法的分裂函数 best_split（S,A）与 C4.5 算法相比，也只是分裂指标不同而已，其余均类似，这里就不再详细介绍了。

2. CART 算法如何停止分裂

CART 算法停止分裂的条件比 C4.5 要丰富得多，分析人员可以通过设定如下的参数来进行前剪枝。

- ❑ 决策树最大深度（maxdepth）：如果决策树的层数已经达到了制定的深度，则停止生长。
- ❑ 最小样本分裂数（minsplit）：如果节点所包含的样本数已低于最小样本数 minsplit，则不再分裂。
- ❑ 树中叶节点所包含的最小样本数（minbucket）：如果自变量 A 分裂后生成的叶节点所包含的样本数低于最小样本数 minbucket，则此次分裂无效，自变量 A 不能再进行分裂。

❑ 复杂度系数最小值 mincp：在决策树的每个决策节点均可计算出一个复杂度系数（由根节点到叶节点不断减小，具体请参考 7.1.3 节），如果本次分裂后的复杂度系数小于 mincp，则不分裂。

3. 后剪枝算法 MCCP

如果在生成决策树时，一味地追求训练集处理结果的最小错误率，则决策树可能会很大，在预测新样本时，其误差也可能会很大，如图 7-8 中虚线的右侧部分所示，此时称为过分拟合（overfitting）。

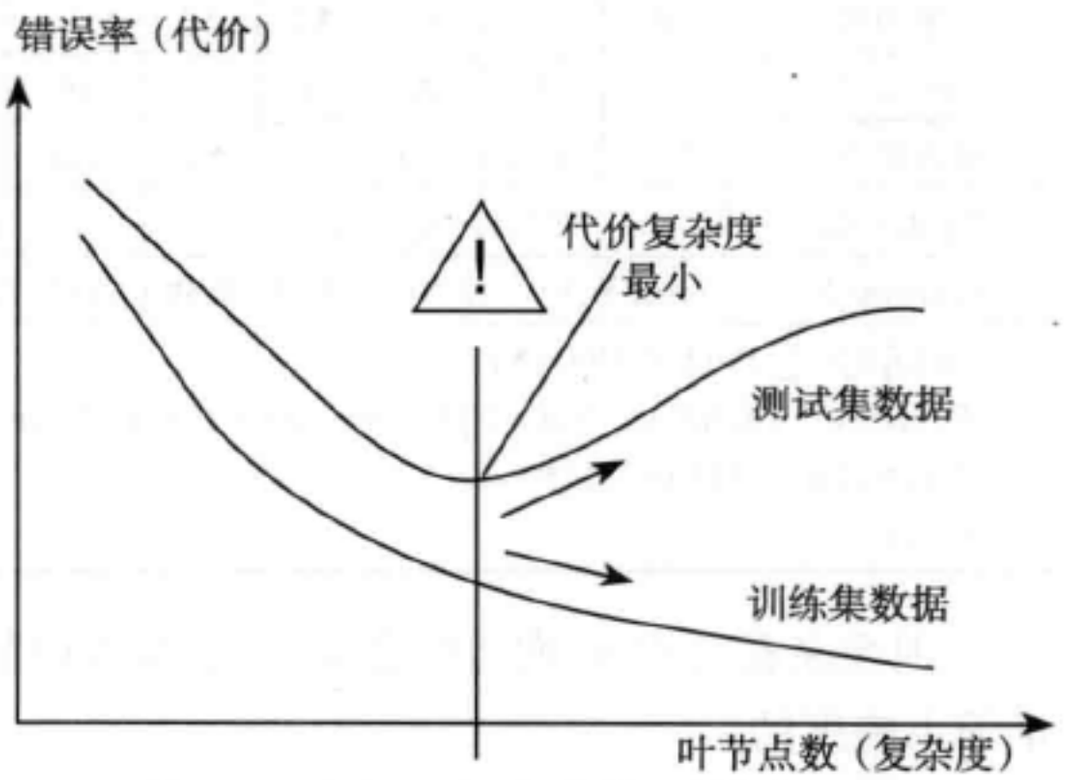


图 7-8　最小代价复杂度剪枝法示意图

CART 的剪枝算法采用的是最小代价复杂度剪枝法（Minimal Cost Complexity Pruning，MCCP）。其目的是使误差和决策树的规模都尽可能地小，即剪除虚线右侧的多余节点。通常，叶节点数与决策树复杂度成正比。如果把决策树的误差视为代价，那么引入决策树 T 的代价复杂度则记为 $R_{cp}(T)$，其定义为：

$$R_{cp}(T)=R(T)+cp*|T_{\text{num}}|$$

其中，$R(T)$ 是决策树 T 在测试集数据上的分类误差，即图 7-8 中红线的 y 轴数据。$|T_{\text{num}}|$ 表示叶节点数。cp 是复杂度系数，表示每增加一个叶节点所带来的代价复杂度。下面介绍如何使用代价复杂度来进行剪枝。

在图 7-9 中，把决策节点 2 记为 $\{t\}$。决策节点 2 下的一个子树记为 T_t，其中包括决策点 3 和叶节点 4、叶节点 5。

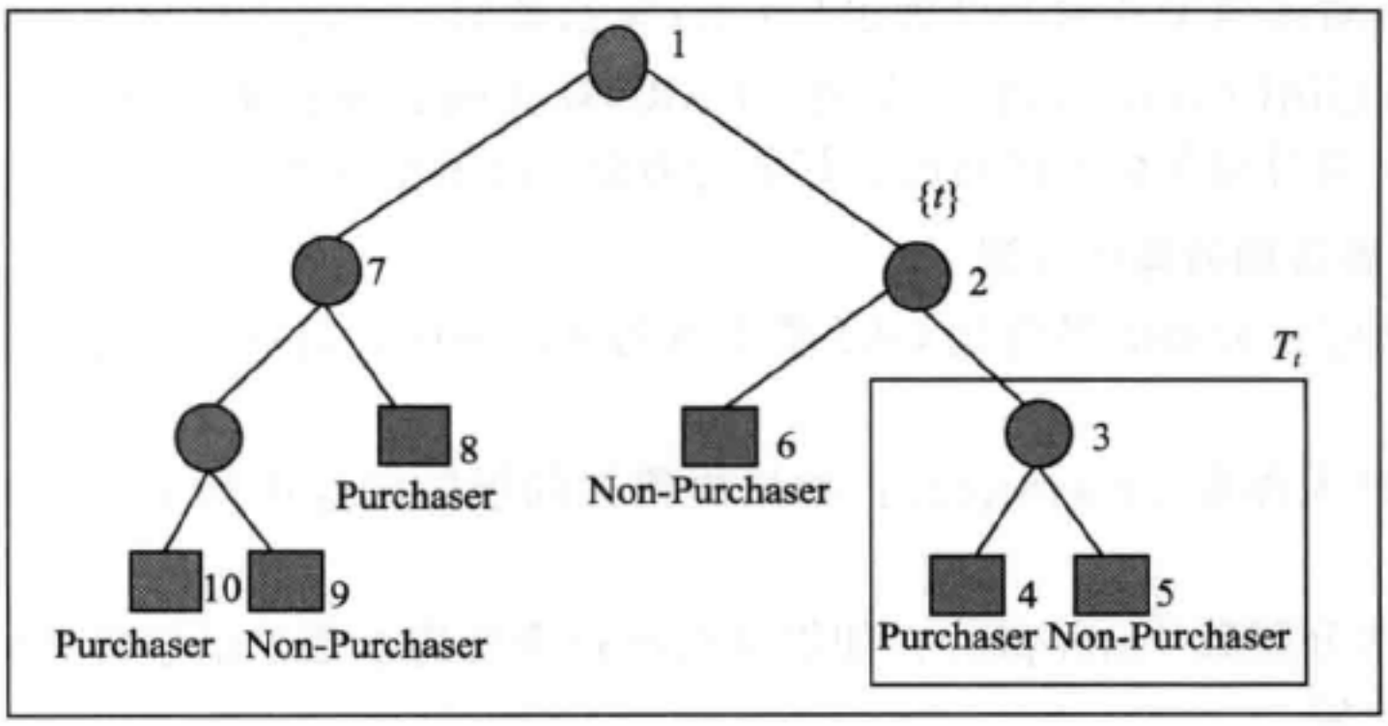


图 7-9　示例图

节点 $\{t\}$ 的代价复杂度就是 $R_{cp}(\{t\})=R(t)+cp$，其中 $R(t)$ 为决策节点 t 上的分类误差，其

子树 T_t 的代价复杂度为 $R_{cp}(T_t)=R(T_t)+cp*|T_{t\text{-num}}|$。如果 $R_{cp}(\{t\}) < R_{cp}(T_t)$，则把子树 T_t 剪除，反之则保留。实际上，每个决策点均可以计算出一个对应的复杂度系数 cp，并且该系数会从叶节点至根节点不断变大。有了使用代价复杂度来进行剪枝的基础，就可以说明最小代价复杂度剪枝法的过程了。具体过程如下：

1）把未做剪枝的复杂度最高的树记为 T_1，并令 cp=0。之后按照上面提到的方法计算各决策节点及其子树的代价复杂度，并逐步增加系数 cp 的值，直到减少到只有一个子树为止，这里得到 T_2。重复上述过程，直到只剩下一个根节点（其复杂度最低）为止，并将其记为 T_k。

2）在树序列 $T_1 \sim T_k$ 中选取其代价复杂度最小的树，记为 $T_{\min}$。此时可以简单地把树 $T_{\min}$ 作为剪枝后的决策树（记为 T_{opt}）。但是此时往往还要考虑 $T_{\min}$ 在训练集上预测误差的标准误差，它记为 $SE(R(T_{\min}))$，计算式为：

$$SE(R(T_{\min}))= (R(T_{\min})*(1- R(T_{\min}))/N)^{1/2}$$

其中 N 是训练集的样本量。最后通过 $R(T_{opt}) \leqslant R_{cp}(T_{\min})+m*SE(R(T_{\min}))$ 来最终确定剪枝后的最佳树 T_{opt}。其中 m 表示需要考虑的标准误差数。

4. 交叉验证

交叉验证（Cross-Validation）方法可以估计不同“剪枝”条件下各模型的误差，并最终选择误差最小的树模型。对于 K 次交叉验证，可把原始数据采样分割成 K 个子样本，其中的一份作为验证模型的数据，其余的 K-1 份样本则用来训练。交叉验证重复 K 次，每个子样本验证一次，平均 K 次的结果，最后得到一个单一估测。CART 算法可以很好地与交叉检验相结合，使建立的模型更具有一般性。

5. 在 R 语言中实现 CART 算法

在 rpart 包中针对 CART 决策树算法提供的函数主要是用于建模型的 rpart 函数，以及用于剪枝的 prune 函数。

（1）rpart 函数

rapart 函数的基本形式为：rpart(formula, data, subset, na.action = na.rpart, method,parms, control, ...)，其参数说明如表 7-11 所示。

表 7-11 rpart 函数参数说明

参数	说明
formula	分类方程形式，例如 $Y \sim X1+X2+X3$，表示使用自变量 $X1$、$X2$、$X3$（连续型或分类型）来对目标变量 Y（只能是分类型，即因子 factor 对象）做决策树模型
data	包含参数 formula 中变量的数据框
subset	用于在 data 中选取部分样本（部分行数据）
na.action	缺失数据的处理方法，默认删除目标变量的数据，保留自变量的数据
method	变量分割方法，该参数有 4 种取值：连续型对应 anova，分类型（因子）对应 class，计数型对应 poisson（泊松），生存分析型对应 exp。一般情况下，函数会自动检验目标变量的数据类型，自动匹配合适的取值。不过最好指明该参数，对于 CART，使用 class
control	模型建立时用于停止分裂的一些参数，其值是 rpart.control 函数的输出对象。这里以对分类型数据进行 CART 算法建模（目标变量为 factor 因子类型数据）为例，说明 rpart.control 函数的主要参数

（续）

参数	说明
minsplit	参考 7.1.3 节中的“最小样本分裂数”的内容，默认为 20
minbucket	参考 7.1.3 节中“树中叶节点所包含的最小样本数”的内容，默认为 minsplit/3 的取整值
cp	参考 7.1.3 节中“复杂度系数最小值 mincp”的内容，默认为 0.01
maxdepth	参考 7.1.3 节中“决策树最大深度”的内容，默认为 30
xval	交叉检验的次数，默认为 10

这里仍以 7.1.2 节中的新书购买相关数据为实例来进行说明，代码如下：

```
library("rpart")
cart.sol<-rpart(mod.formula,data=data.train,control= rpart.control(minbucket
=20, maxdepth=10,xval=10)); cart.sol
    n= 5127
    node), split, n, loss, yval, (yprob)
          * denotes terminal node
      1) root 5127  2550  0  (0.5026331 0.4973669)
        2) ArtBks< 1.5  4685  2236  0  (0.5227321 0.4772679)
          4) R>=7  3451  1511  0  (0.5621559 0.4378441)
            8) GeogBks< 0.5  2321   905  0  (0.6100819 0.3899181) *
            9) GeogBks>=0.5  1130   524  1  (0.4637168 0.5362832)
              18) M>=421.5  29  0  0  (1.0000000 0.0000000) *
             < 以后省略 >
```

上述代码使用了 10 折交叉验证，并规定决策树的最大层级深度为 10，叶节点的最小样本数为 20。得到的结果如下：

在 5127 个样本中，2550 个目标变量的值为 1（购买了），另外 2577 个的值为 0（未购买）。

在上述代码中，节点 8 的解释如图 7-10 所示。

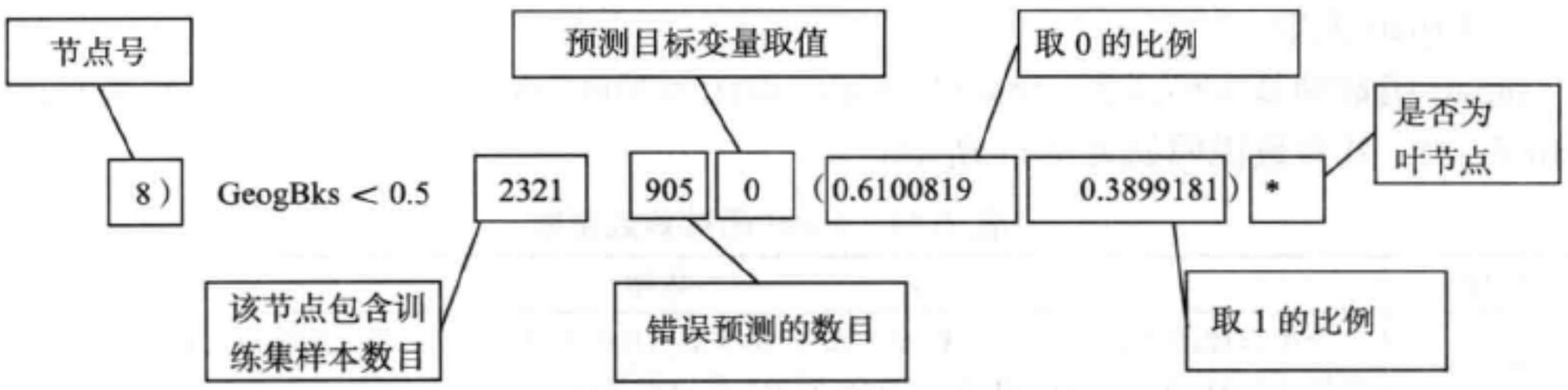


图 7-10　节点详述图

当 ArtBks< 1.5、R>=7 且 GeogBks< 0.5 时，包括 2321 个样本数据，预测目标变量的值为 0（未购买）。其中 905 个样本预测错误（值为 1）。值为 0 的样本占整体的比例为 0.6100819，值为 1 的比例为 0.3899181，错误比例为 38.99%（值为 1 的比例）。该规则的后行有星号，表示它是叶节点。其余节点均可以参考节点 8 进行解释。

查看模型的 cp 等相关信息。

```
cart.sol$cp
          CP nsplit  rel error    xerror       xstd
1 0.07882353      0  1.0000000   1.0156863   0.01403907
2 0.03215686      2  0.8423529   0.8611765   0.01389480
3 0.01137255      3  0.8101961   0.8160784   0.01378889
4 0.01078431      4  0.7988235   0.8129412   0.01378042
5 0.01019608      6  0.7670588   0.8082353   0.00776744
6 0.01000000      7  0.7772549   0.8169019   0.01379963
```

其中 cp 是每次分裂对应的复杂度系数（参考 7.1.3 节）。例如，第 2 行的 cp 的值为 0.03215686，表明这是在进行第 2 次分裂后（nsplit=2）的复杂度系数。可以通过增大 rpart.control 函数中的 cp 参数取值来对树进行前剪枝。

在 summary(cart.sol) 结果中显示如下内容：

```
          <省略头部>
Node number 8: 2321 observations
  predicted class=0  expected loss=0.3899181
    class counts:  1416   905
   probabilities: 0.610 0.390
          <省略尾部>
```

它表明叶节点 8 包含 2321 个样本，其中模型结果（predicted class）为 0，错误率（expected loss）为 0.3899181。0 和 1 的样本数为 1416 和 905，比例为 0.610 ： 0.390。

（2）prune 函数

prune 函数可以实现最小代价复杂度剪枝法，其基本的数学原理可以参考 7.1.3 节。在由 rpart 函数计算得到的 cart 决策树中，每个节点均输出一个对应的 cp，prune 函数通过设置 cp 参数来对决策树进行剪枝。

首先使用第 6 章的 1− 标准差规则找到最佳剪枝 cp ： xerror 的最小值（为 nsplit=6 时的 0.8082353）对应的 xstd=0.01376744。然后选取 xerror 在 0.8082353 ± 0.00776744 范围（即 [0.8004679, 0.8160027]）内 cp 的最小值作为最合适的复杂度系数 cp。此时符合该条件的是 nsplit 等于 4 和 6 时的 cp，其中 nsplit=6 时的 cp 较小，所以要剪除 nsplit=7 时产生的分支。这里取剪枝的最佳 cp 值为 0.0102，并通过 prune 函数来进行剪枝处理。代码如下：

```
cart.sol.pru<-prune(cart.sol,cp=0.0102)
```

使用如下代码观察剪枝后的 cp 值。

```
cart.sol.pru$cp
          CP nsplit rel error    xerror       xstd
1 0.07882353      0 1.0000000 1.0156863 0.01403907
2 0.03215686      2 0.8423529 0.8611765 0.01389480
3 0.01137255      3 0.8101961 0.8160784 0.01378889
4 0.01078431      4 0.7988235 0.8129412 0.01378042
5 0.01020000      6 0.7772549 0.8090196 0.01376963
```

可见剪枝后的树只分裂了 6 次（<7）。

最后绘制 cart.sol.pru 的树状图。代码如下：

```
plot(cart.sol.pru)
text(cart.sol.pru,all=TRUE,digits=7,use.n=TRUE,cex=0.9,xpd=TRUE)
```

绘制结果如图 7-11 所示。

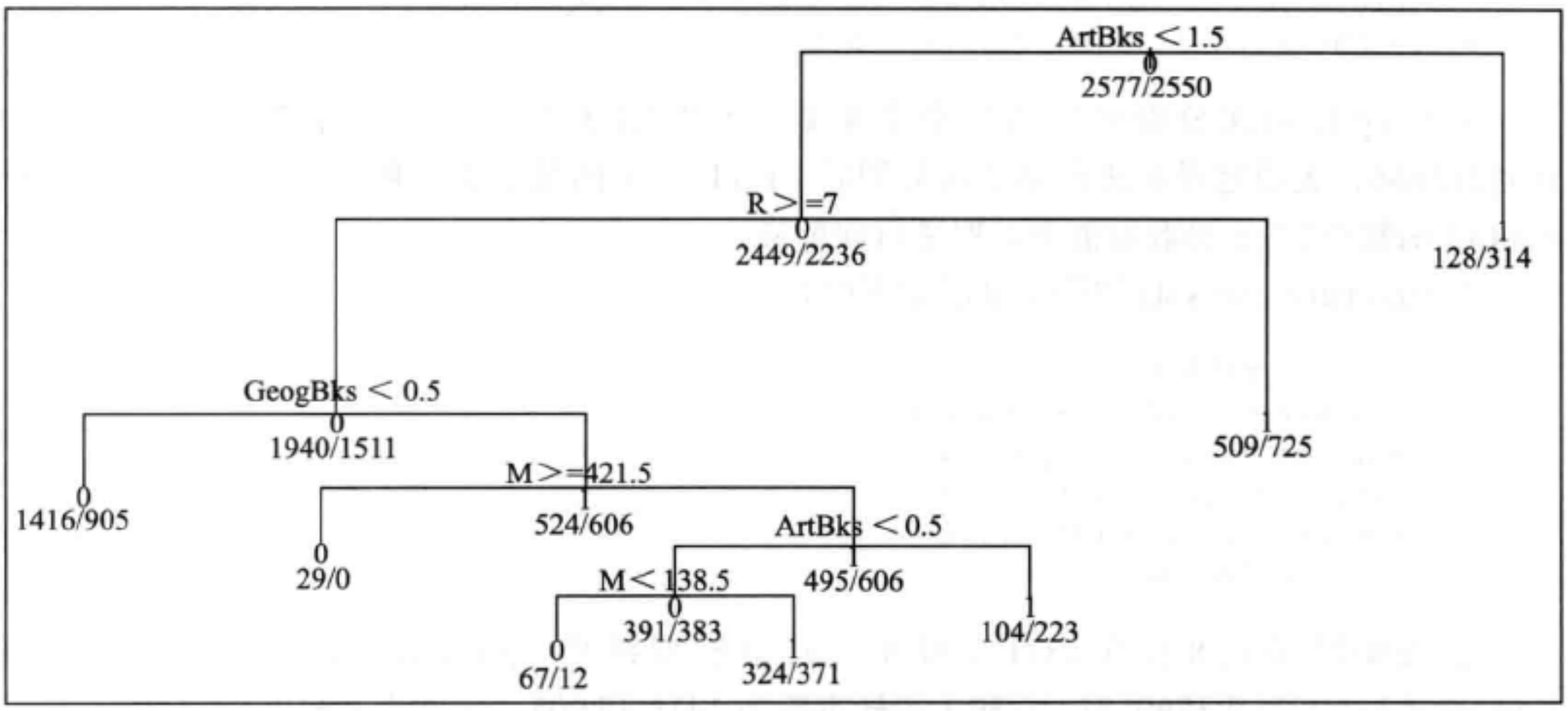


图 7-11　分类树模型的树状图

6. 预测

在对 rpart 函数产生的模型进行预测时仍会使用 predict 函数，其中涉及的大部分参数可以参考 7.1.2 节的内容。这里先给出其中的不同之处，即正常情况下 predict 函数返回取值为 0 和 1 的概率：

```
cart.pred<-predict(cart.sol,data.test); cart.pred[1:2,]
             0         1
3846   0.6100819   0.3899181
3267   0.6100819   0.3899181
```

其中，输出结果对象 cart.pred 的第一列 cart.pred[,1] 是目标变量为 0 的概率，第二列 cart.pred[,2] 是目标变量为 1 的概率。可以通过一个单位阶跃函数得到最后的预测分类。代码如下：

```
cart.pred.factor <-ifelse(cart.pred[,2]>0.5,1,0); cart.pred.factor[1:2]
3846 3267
   0    0
```

上述代码也可以通过设置 predict 函数的 type 参数为“class”来实现。代码如下：

```
cart.pred <- predict(cart.sol,data.test,type="class"); cart.pred[1:2]
3846   3267
   0      0
Levels: 0 1
```

注意，返回的 cart.pred 不再是一个数据框，而是一个 factor 向量。

注意

CART 算法的特点为：

输入变量（自变量）：为分类型变量或连续型变量。

输出变量（目标变量）：为分类型变量（或连续型：回归分析）。

连续变量处理：N 等分离散化。

树分枝类型：二分枝。

分裂指标：gini 增益（分裂后的目标变量取值变异较小，纯度高）。

前剪枝：参见 7.1.3 节。

后剪枝：使用最小代价复杂度剪枝法。

7.1.4 条件推理决策树算法

条件推理决策树算法（Conditional inference trees）的分裂方式不再以自变量分裂后目标变量的纯度（如 C4.5 和 CART 算法）为分裂度量指标，而是以自变量与目标变量的相关性（一些统计检验）为分裂度量指标。卡方自动交叉（Chi-squared Automatic Interaction Detector，CHAID）算法是最早期的条件推理决策树，它使用卡方独立性检验（对于分类型自变量）和方差分析（连续型自变量）来衡量相关性。后来又发展出了快速无偏有效统计树（Quick Unbiased Efficient Statistical Tree，QUEST）算法。本章以 party 包中的 ctree 函数为核心进行说明。

1. 基本原理

首先，假设所有自变量与目标变量均是独立的，再对它们进行独立性检验。如果独立性检验 P 值小于设置的阈值（有显著性差别，即该自变量对目标变量有影响），则该自变量可以作为决策树节点。之后计算各自变量与目标变量间的相关性，并选择相关性最强的自变量作为本次分裂的决策点。决策点变量选择好后，用置换检验（Permutation Test）来选择分割点。对于剪枝，由于独立性检验 P 值的阈值已经决定了模型的复杂程度，所以不必再对其进行特殊剪枝，只要选择适当的阈值即可。

2. 在 R 语言中实现条件推理决策树

R 语言的 party 包主要用于实现条件推理决策树，其核心函数是 ctree，参数说明见表 7-12。

表 7-12 ctree 函数参数说明

参数	说明
data	包含 formula 参数中变量的数据框
control	模型建立时用于停止分裂的一些参数，其值是 ctree_control 函数的输出对象。这里以对分类型数据进行建模（目标变量为 factor 因子类型数据）为例，说明 ctree_control 函数的主要参数

（续）

参数	说明
teststat	独立性检验 P 值的处理方式："quad"、"max"
testtype	独立性检验方式："Bonferroni"、"MonteCarlo"、"Univariate"、"Teststatistic"
mincriterion	默认值为 0.95。7.1.4 节中的独立性检验 P 值的阈值等于 1- mincriterion。例如，如果设置 mincriterion=0.95，则只有独立性检验 P 值小于 0.05（=1-0.95）的自变量才可以成为决策树中的节点。mincriterion 越接近 0，决策树生长得越大
minsplit	最小样本分裂数（与 CART 算法中 rpart_control 函数的 minsplit 相同）
minbucket	叶节点的最小样本数（与 CART 算法中 rpart_control 函数的 minbucket 相同）
maxdepth	决策树最大深度（与 CART 算法中 rpart_control 函数的 maxdepth 相同）

另外，关于 formula data weights subset 参数的说明，请参考 CART 算法的 rpart 函数（见 7.1.3 节）。

这里仍以 7.1.2 节中购书案例为实例：

```
library("party")
party.sol<-ctree(mod.formula,data=data.train,control= ctree_control(mincriterion
=0.99))
party.sol
                    Conditional inference tree with 25 terminal nodes
Number of observations:  5127
1) ArtBks <= 1; criterion = 1, statistic = 123.795
  2) R <= 6; criterion = 1, statistic = 57.191
    3) Gender == {1}; criterion = 1, statistic = 19.14
      4) F <= 9; criterion = 0.999, statistic = 15.854
        5)*  weights = 648
           < 以下忽略 >
```

因为上述语句把 mincriterion 参数设置为 0.99，所以只有独立性检验 P 值小于 0.01(=1-0.99) 的自变量才能成为决策树中的节点。在显示的结果中，criterion 等于 1- 独立性检验 P 值，weights 是节点的样本数。使用 plot 函数直接输出决策树的图形。代码如下：

```
plot(party.sol )
```

输出的图形如图 7-12 所示。

注意

在图 7-12 中，决策节点中的 p 实际是独立性检验 P 值，即 1-criterion。

3. 预测

在对 ctree 函数产生的模型进行预测时仍会使用 predict 函数，默认情况下输出 0 或 1 的 factor 向量：

```
party.pred<-predict(party.sol,data.test); party.pred[1:2]
[1] 0 0
Levels: 0 1
```

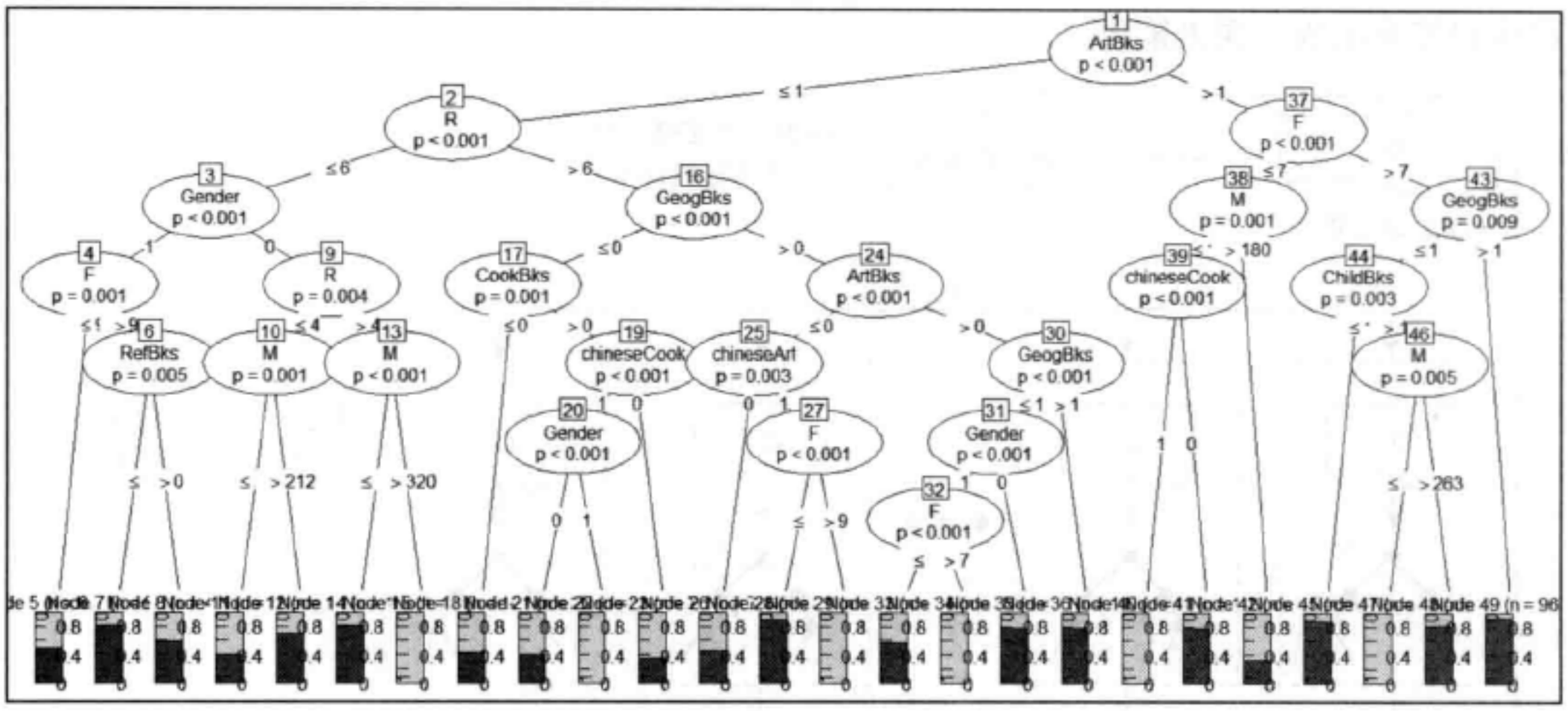


图 7-12 分类树模型的树状图

也可以通过设置 type 参数为 prob 来获取预测为 0 的概率和为 1 的概率。代码如下：

```
party.pred<-predict(party.sol,data.test,type= "prob");
party.pred[[1]]
[1] 0.575 0.425
party.pred[[2]]
[1] 0.6462766 0.3537234
```

注意，此时返回的 party.pred 是一个 list 对象，其读取方式和数据框略有不同。由于测试样本 1 预测为 0 的概率是 party.pred[[1]][1]>0.5，所以实际预测时取值为 0。

注意

条件推理决策树算法的特点为：

输入变量（自变量）：为分类型变量或连续型变量。

输出变量（目标变量）：为分类型变量（或连续型：回归分析）。

连续变量处理：N 等分离散化。

树分枝类型：二分枝（以 party 包中的 ctree 函数为例）。

分裂指标：独立性检验和相关性（分裂后自变量与目标变量的相关性）。

7.1.5 随机森林算法

随机森林（random forest）算法是一种专门为决策树分类器设计的优化方法。它综合了多棵决策树模型的预测结果，其中的每棵树都是基于随机样本的一个独立集合的值产生的。

随机森林的一般步骤如图 7-13 所示。首先基于固定概率分布，从原始训练集中可重复地选取 N 个样本形成 t 个子训练集，然后使用这 t 个训练集产生 t 棵决策树。最后把这 t 棵

决策树综合组成一棵决策树。

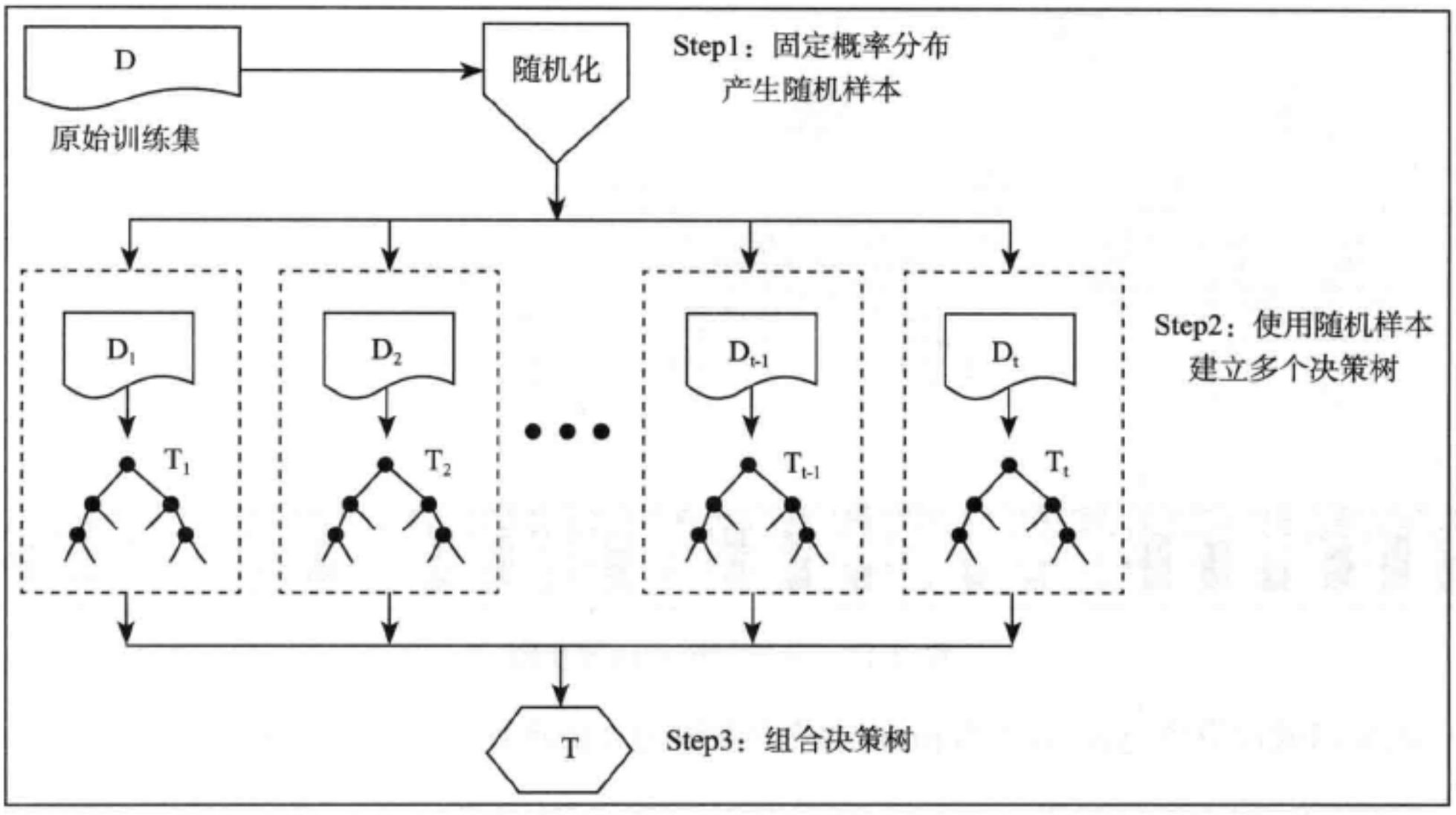


图 7-13 随机森林基本原理

在 R 语言中，基于 CART 算法的随机森林算法所涉及的函数是 rpart 包的 randomForest 函数，基于条件推理决策树算法的随机森林算法所涉及的函数是 party 包的 cforest 函数。函数的使用方法可参考 7.1.3 节和 7.1.4 节的 rpart 和 ctree 函数。

一般来说，随机森林算法的效果要比一般的决策树均好很多。

7.2 贝叶斯分类

贝叶斯分类器（Bayesian classifiers）是基于贝叶斯定理计算目标变量条件概率 $P\{Y|X\}$（自变量在不同取值下，目标变量各分类取值的概率）的一种建模方法。在 R 语言中，贝叶斯统计分析以及分类器的包很丰富，这里主要介绍朴素贝叶斯算法的实现方式：e1071 包中的 naiveBayes 函数。

7.2.1 贝叶斯定理

1. 基本定理概述

假设有随机变量 X 和 Y，$P\{X=x\}$ 表示变量 X 的值为 x 时的概率，$P\{Y=y\}$ 表示变量 Y 取 y 值时的概率，$P\{X=x,Y=y\}$ 表示变量 X 的值为 x 且为 Y 的值为 y 时的联合概率，$P\{Y=y \mid X=x\}$ 表示在 X 的值为 x 的前提下，Y 的值为 y 的条件概率。图 7-14 为随机变量 X 和 Y 在值为 x 或 y 时的样本数据。

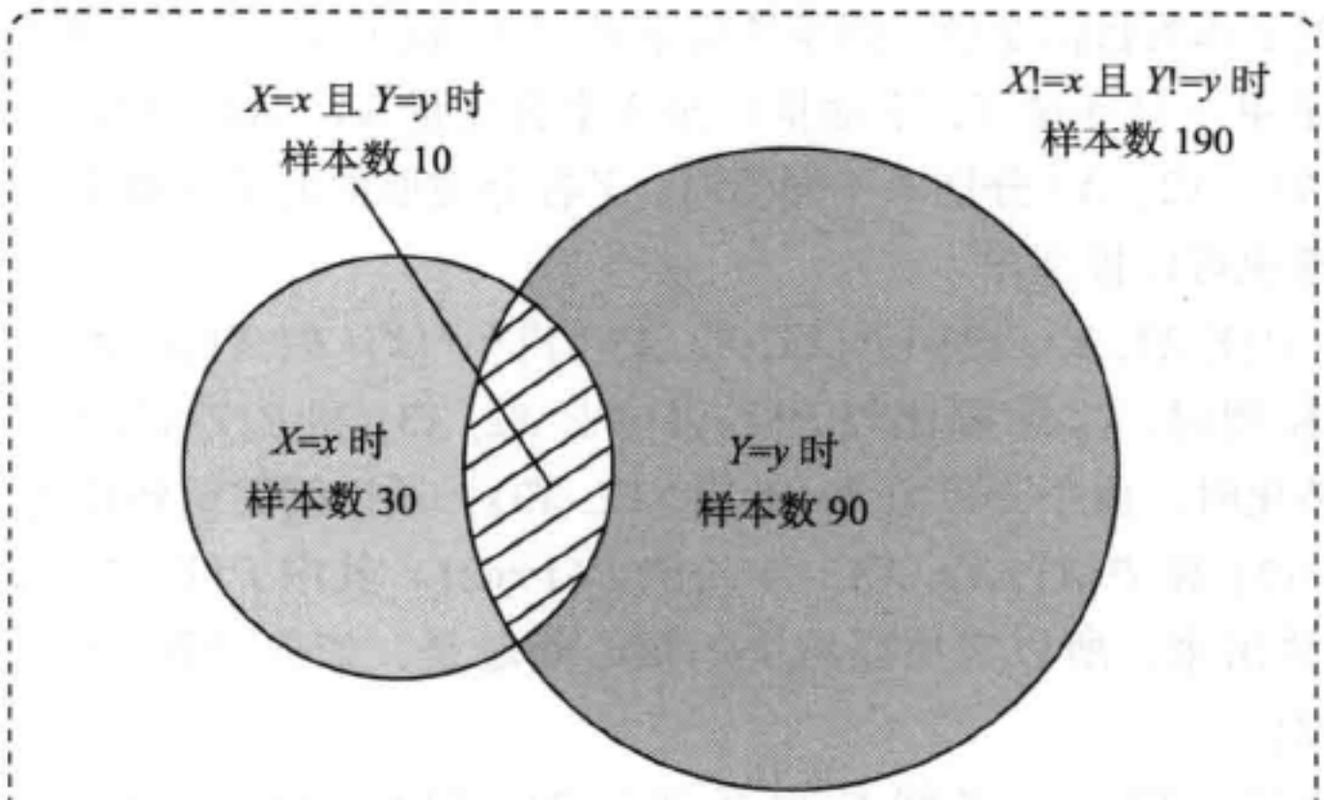


图 7-14 样本数据

其整体的样本空间数为 190+30+90-10=300，其计算过程如表 7-13 所示。

表 7-13 整体样本空间数的计算过程

变量 X、Y 的值	样本数	概率
$X=x$	30	$P\{X=x\}=30/300$
$Y=y$	90	$P\{Y=y\}=90/300$
$X=x$ 且 $Y=y$	10	$X=x$ 且 $Y=y$ 的联合概率 $P\{X=x,Y=y\}=10/300$
		$X=x$ 的前提下，$Y=y$ 的条件概率 $P\{Y=y \mid X=x\}=10/30$
		$Y=y$ 的前提下，$X=x$ 的条件概率 $P\{X=x \mid Y=y\}=10/90$

$P\{X=x,Y=y\}= 10/300$

$=P\{Y=y \mid X=x\}* P\{X=x\}=10/30*30/300=10/300$

$= P\{X=x \mid Y=y\}* P\{Y=y\}=10/90*90/300=10/300$

由表 7-13 可以总结出贝叶斯定理，如下：

$$P\{X=x,Y=y\}= P\{Y=y \mid X=x\}* P\{X=x\}= P\{X=x \mid Y=y\}* P\{Y=y\}$$

把 Y 和 X 的值推广至所有分类值，则可以写为

$$P\{X,Y\}= P\{Y \mid X\}* P\{X\}= P\{X \mid Y\}* P\{Y\} \quad (7\text{-}1)$$

2. 贝叶斯定理的分类器应用

如果想使用一个三元自变量 X 来对二元目标变量 Y 进行分类器建模，则可通过训练集计算出各条件概率，并比较其大小来实现，如表 7-14 所示。

表 7-14 预测目标变量

	Y=y1	Y=y2	根据条件概率建立的分类器模型
$X=x1$	$P\{Y=y1 \mid X=x1\}=0.4$	$P\{Y=y2 \mid X=x1\}=0.6$	如果自变量 X 的值为 $x1$，则 Y 的值为 $y2$
$X=x2$	$P\{Y=y1 \mid X=x2\}=0.3$	$P\{Y=y2 \mid X=x2\}=0.7$	如果自变量 X 的值为 $x2$，则 Y 的值为 $y2$
$X=x3$	$P\{Y=y1 \mid X=x3\}=0.8$	$P\{Y=y2 \mid X=x3\}=0.2$	如果自变量 X 的值为 $x3$，则 Y 的值为 $y1$

可见，如果把 Y 作为目标变量，则条件概率 $P\{Y|X\}$ 就可以作为分类器的核心度量尺度。

上面只考虑了单一自变量 X，下面推广至 3 个自变量 $X1$、$X2$、$X3$。此时分类器的核心度量为：自变量 $X1$、$X2$、$X3$ 分别取不同值时，Y 各分类取值的条件概率为 $P\{Y|X1,X2,X3\}$。同样由贝叶斯定理也可以推理出

$$P\{Y|X1,X2,X3\}=P\{X1,X2,X3\mid Y\}*P\{Y\}/P\{X1,X2,X3\} \qquad (7\text{-}2)$$

在建立分类模型时，实际要比较 $P\{Y=y1|X1,X2,X3\}$ 和 $P\{Y=y2|X1,X2,X3\}$ 的大小。使用式 7-2 进行转化时，由于分母恒为 $P\{X1,X2,X3\}$，所以问题就转化为了比较 $P\{X1,X2,X3\mid Y=y1\}*P\{Y=y1\}$ 和 $P\{X1,X2,X3\mid Y=y2\}*P\{Y=y2\}$。其中 $P\{Y=y1\}$ 和 $P\{Y=y2\}$ 可以通过训练集快速估算出来，所以实质要解决的核心问题是：如何计算 $P\{X1,X2,X3|Y=y1\}$ 和 $P\{X1,X2,X3|Y=y2\}$。

假设自变量 $X1$、$X2$、$X3$ 之间是相互独立的，$P\{X1,X2,X3|Y\}=P\{X1|Y\}*P\{X2|Y\}*P\{X2|Y\}$，则式 7-2 可以转化为

$$P\{Y|X1,X2,X3\}=P\{X1|Y\}*P\{X2|Y\}*P\{X2|Y\}*P\{Y\}/P\{X1,X2,X3\}$$

如果推广至 n 个自变量，并假设自变量间互相独立，则可得到朴素贝叶斯分类器的基本公式：

$$P\{Y|X1,...,Xn\}=(\prod\nolimits_n P\{X_i|Y\})*P\{Y\}/P\{X1,...,X3\} \qquad (7\text{-}3)$$

有了互相独立的假设后，就不必计算各个自变量每一个组合的概率 $P\{X1,...,Xn|Y\}$ 了，只需要在给定目标变量 Y 后计算每个自变量 Xi 的概率 $P\{Xi|Y\}$ 即可。这样一来，不必使用很大的训练集样本就可以得到较好的概率估计效果。

7.2.2　朴素贝叶斯分类器

前面介绍了朴素贝叶斯分类公式的基本推理过程。朴素贝叶斯算法不仅可以处理分类型自变量，还可以处理连续型变量，其原理这里暂不介绍。在 e1071 包中，可使用 naiveBayes 函数通过朴素贝叶斯算法来建立分类器模型。这里使用 7.1.2 节的购书数据集作为朴素贝叶斯分析的数据，其代码如下：

```
library("e1071")
nb.sol<-naiveBayes(mod.formula,data=data.train);nb.sol
Naive Bayes Classifier for Discrete Predictors
Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)
A-priori probabilities:
Y
        0         1
0.4956115 0.5043885
Conditional probabilities:
   Gender
Y           0         1
  0 0.2786305 0.7213695
  1 0.3785770 0.6214230
```

```
   M
Y       [,1]     [,2]
  0 206.6482 101.6138
  1 218.1036 104.3643
       <中间部分暂不展示>
   chineseArt
Y            0         1
  0 0.95828414 0.04171586
  1 0.91492653 0.08507347
```

在上面的结果中，A-priori probabilities 实际是公式 7-3 中的 $P\{Y\}$。Conditional probabilities 实际是公式 7-3 中的 $P\{Xi|Y\}$。通过以上数据即可快速比较 $P\{Y|\ X1,...,Xn\}$ 的大小，从而建立出分类模型。

```
nb.sol.pre<-predict(nb.sol, newdata=data.test)
table(nb.sol.pre, data.test$HanDynastyArt)
nb.sol.pre   0     1
         0   828   620
         1   293   456
```

predict 函数可以使用建立好的模型 sol 来预测新数据样本。在上面的代码中，把新数据参数 newdata 设置为测试集数据 data.train，nb.sol.pre 就是对测试集的预测，table(nb.sol.pre, data.test$HanDynastyArt) 则展示了混淆矩阵。

由于朴素贝叶斯算法的假设是建立在自变量之间互相独立的基础上的（这个假设较为严苛），而在更多情况下，部分变量间是存在关联的，在这种情况下，要通过建立贝叶斯网络分析来建立分类器模型，这里暂不介绍。

7.3 支持向量机 SVM

7.3.1 原理

1. 超平面

首先引入一个虚拟数据集 data，其格式见表 7-15。

表 7-15 数据集

id	$x1$	$x2$	y
1	7.284173	40.08300	0
2	8.779236	41.21867	0
…	…	…	…
250	39.5006	41.99453	1

其中，$x1$ 和 $x2$ 是连续型变量，目标变量 y 是分类型变量，其值为 0 和 1。下面绘制 $x2 \sim x1$ 平面散点图，并把目标变量 y 为 0 的样本点置为蓝色，为 1 的点置为红色，如图 7-15 所示。

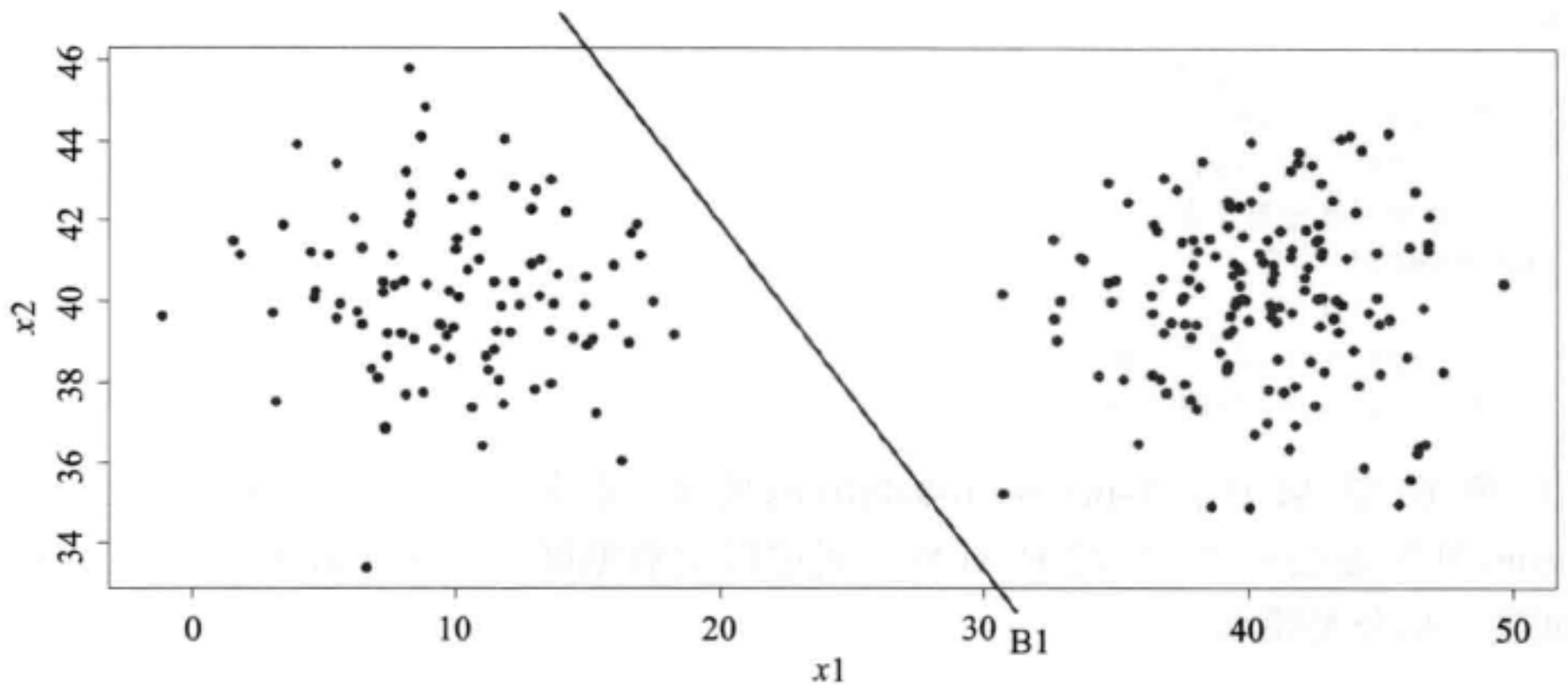


图 7-15　示例图

如果在该 $x2 \sim x1$ 平面内绘制一条线性的直线（图 7-15 中的 B1），把两类样本点分隔开，那么这条直线是一个超平面。注意，之所以称其为平面，是因为如果自变量数大于 2，则不能只用一条直线来分隔，而是要使用一个多维度的面来分隔。

2. 超平面的边缘

在 $x2 \sim x1$ 平面内可以找到多条类似于 B1 直线的超平面（见图 7-16），为了挑选出其中最优的一个超平面，引入一个新的概念：超平面的“边缘”。

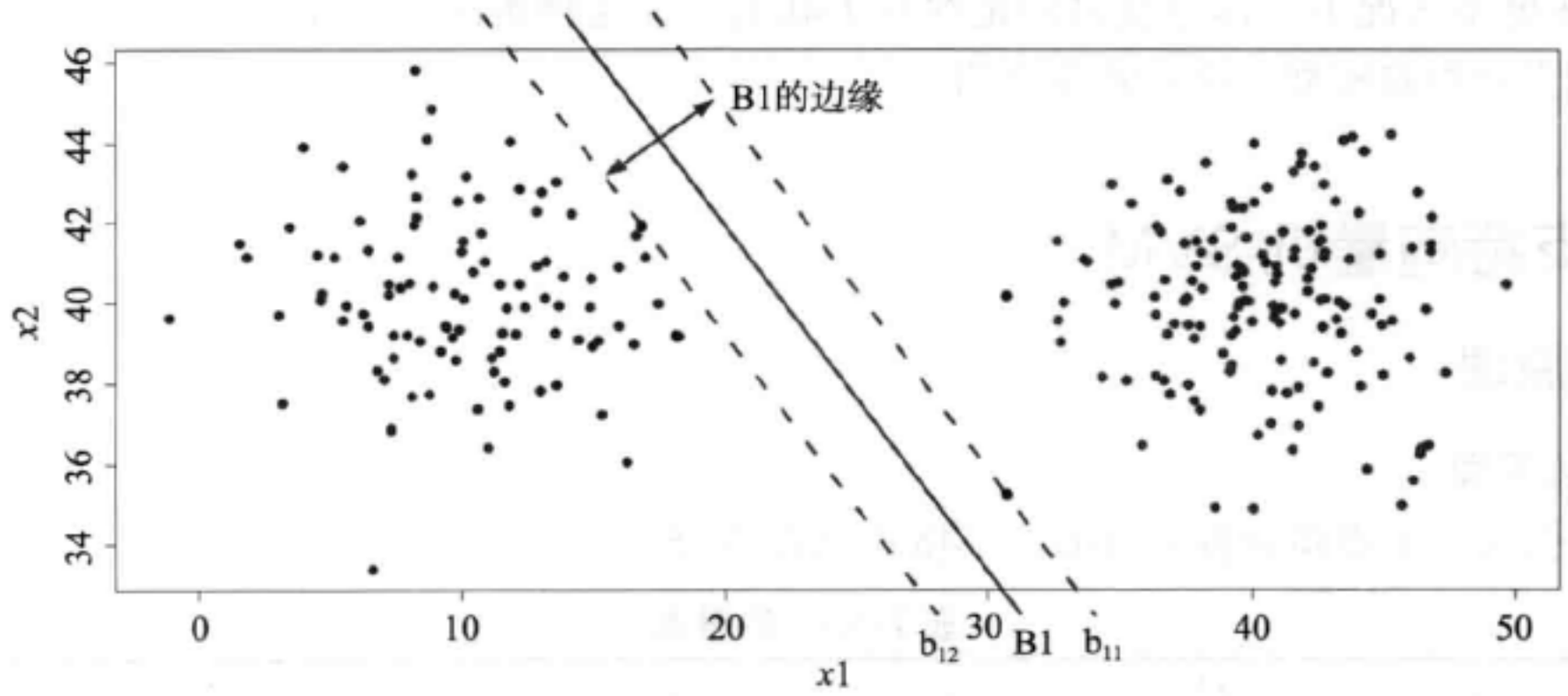


图 7-16　超平面示例图

通过超平面 B1 可以找到一对超平面 b_{11} 和 b_{12}，其中 b_{11} 是通过“把 B1 平移，直到接触到红色样本点为止”而得到的，类似的 b_{12} 是平移后接触到蓝色样本点为止得到的。b_{11} 和 b_{12} 之间的距离就是超平面 B1 的边缘。

3. 最大边缘超平面

下面比较两个超平面 B1 和 B2 的边缘，如图 7-17 所示。

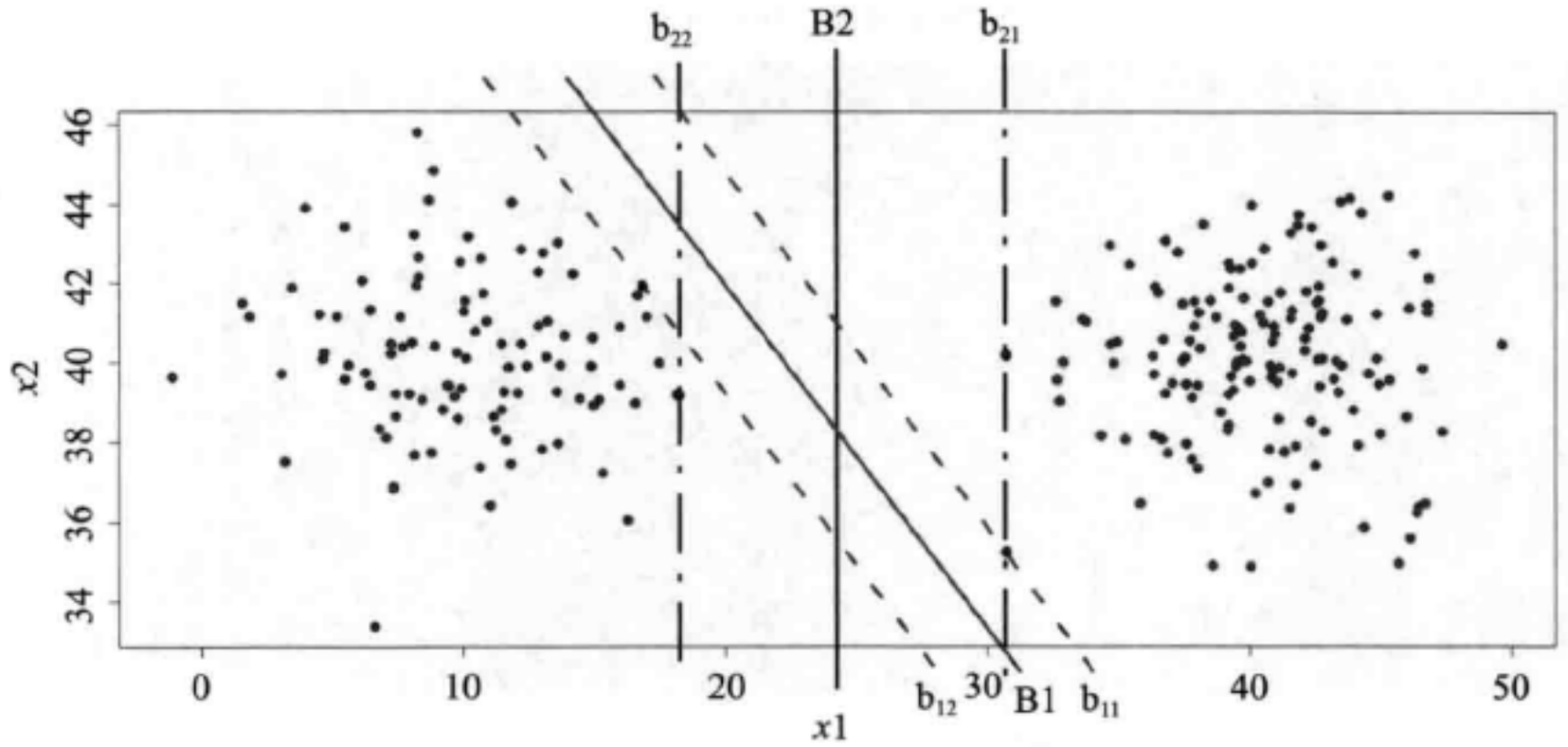


图 7-17 最大超平面示例图

很明显，B2 的边缘大于 B1 的边缘，在这种情况下，可以认为 B2 更适合于做 $x2 \sim x1$ 的超平面，即作为分类器 $y=f(x1,x2)$ 的划分规则。Svm 分类器算法就是用于找到这个合适的超平面 B2 的。

注意

如果考虑使用直线作为超平面，则称其为线性 SVM，而如果使用曲线作为超平面，则称其为非线性 SVM。

7.3.2 在 R 语言中实现非线性 SVM 分析

在 R 语言中使用 e1071 包的 svm 函数来实现非线性 svm 分析。下面对下列节前面虚拟的 data 数据集进行 svm 分析，注意 data$y 作为目标变量必须设置为 factor 分类类型的数据。代码如下：

```
library(e1071)
svm.sol<-svm(y ~ x1+x2,data=data, probability=TRUE)
plot(x=svm.sol,data=data,formula=x2 ~ x1)
```

把参数 formula 设置为 $y \sim x1+x2$，参数 data 设置为 data，表示在 svm 分析中，自变量为 data$x1 和 data$x2，目标变量为 data$y。将参数 probability 设置为 TRUE，则输出结果是目标变量为 0 和 1 的概率。

最后使用 plot 函数绘制最优的超平面，如图 7-18 所示。

对 svm.sol 模型的预测可以使用 predict 函数来完成。代码如下：

```
pred.1<-predict(svm.sol,data.test)
pred.2<-predict(svm.sol,data.test, probability=T)
attr(pred.2,"probabilities")[,2]
```

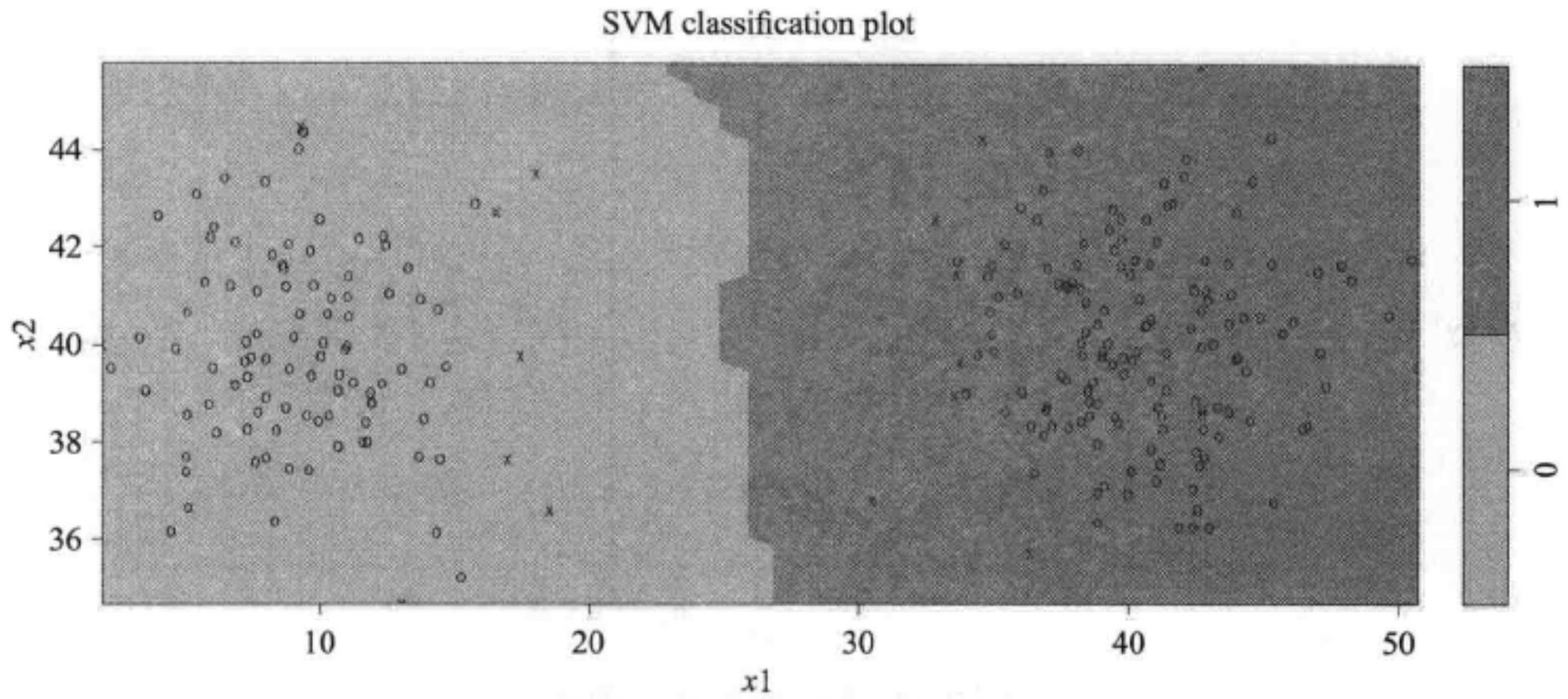


图 7-18　结果展示图

如果参数 probability 设置为 FALSE（默认），则 predict 函数只返回预测的目标变量的 factor 值。例如，上例中 pred.1 返回的是包含 factor0 和 1 的分类向量。如果参数 probability 设置为 TRUE，则 predict 函数会计算预测目标变量取 0 和 1 的概率，并使用 attr(pred.2,"probabilities")[,2] 返回预测的目标变量取 1 的概率。

7.4　人工神经网络

神经网络起源于生物学中有关神经元的研究。神经元通过控制自身的神经突触接受信息。这些神经突触的连接又被认为是大脑行为的关键因素。所谓人工神经网络，就是使用一套函数模型模拟这些相互连接的神经元。它既可以对连续型目标变量做回归分析，也可对分类型目标变量做分类分析，本章主要讲分类分析。

7.4.1　神经元

神经元是神经网络中的基本单元，其结构如图 7-19 所示，主要分为权重系数、加法器、激活函数 3 个部分。

下面是针对上面 3 个部分的说明。

- 带有权重系数 w_j 的突触，连接到输入值 x_j，其中 j= 1,2 , ⋯, m。
- 加法器将所有的输入做加权，并与一个偏差 bias 求和，记为 v。公式如下：

$$v=\text{bias}+\sum_{j=1}^{m}\omega_j x_j$$

可以把偏差 bias 看作是输入 x_0 和权重恒为 1 的系数 w_0 之积，即 bias= $w_0 \times x_0$。此时上式便可改为：

$$v=\sum_{j=0}^{m}\omega_j x_j$$

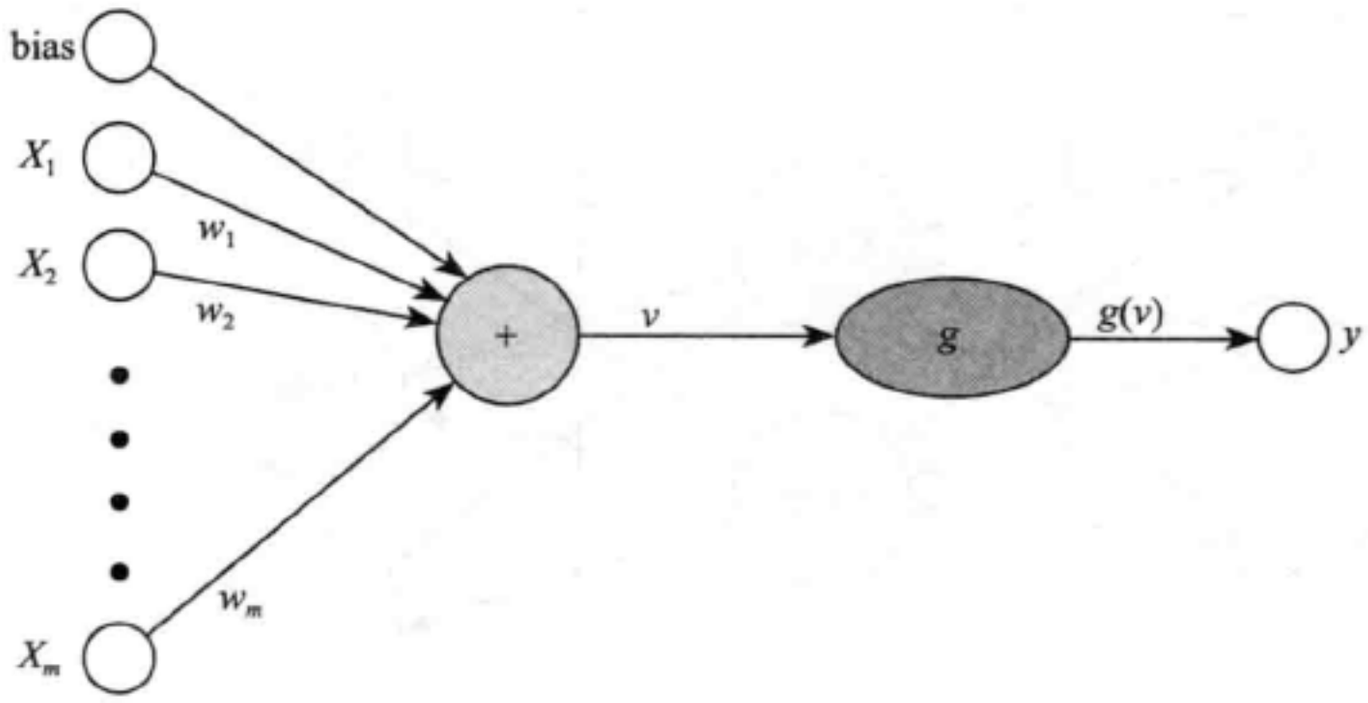
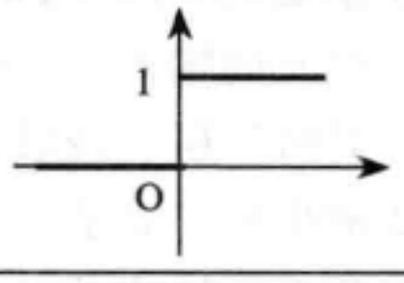
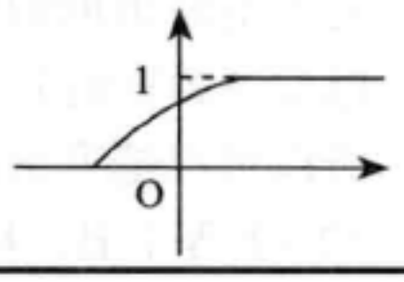


图 7-19 神经元结构图

❑ 激活函数 g（也称为压缩函数），并把 $g(v)$ 作为神经元中的输出，该函数必须是单调函数。常见的激活函数如表 7-16 所示。

表 7-16 激活函数

激活函数	定义	图型
阶跃函数（也称为阈值型函数）	$f(x)=\begin{cases}1 & (x>0)\\ 0 & (x\leqslant 0)\end{cases}$	
Logistic 函数（也称为 Sigmoid 型函数）	$f(x)=\dfrac{1}{1+e^{-x}}$	

实现激活函数为 Logistic 函数的神经元代码如代码清单 7-1 所示。

代码清单 7-1

```
ann.cell<-function(x,w,bias){
    v<-sum(w*x)+bias
    o<-1/(1+exp(-v))
    o
}
```

7.4.2 两层网络

人工神经网络可分为输入层、隐藏层和输出层 3 个部分。假设要使用 3 个自变量对 1 个三元分类变量（取值为 A、B、C）进行两层神经网络分析，并把隐藏层中的神经元数设为 2，如图 7-20 所示，那么这 3 个层面实现的形式如下：

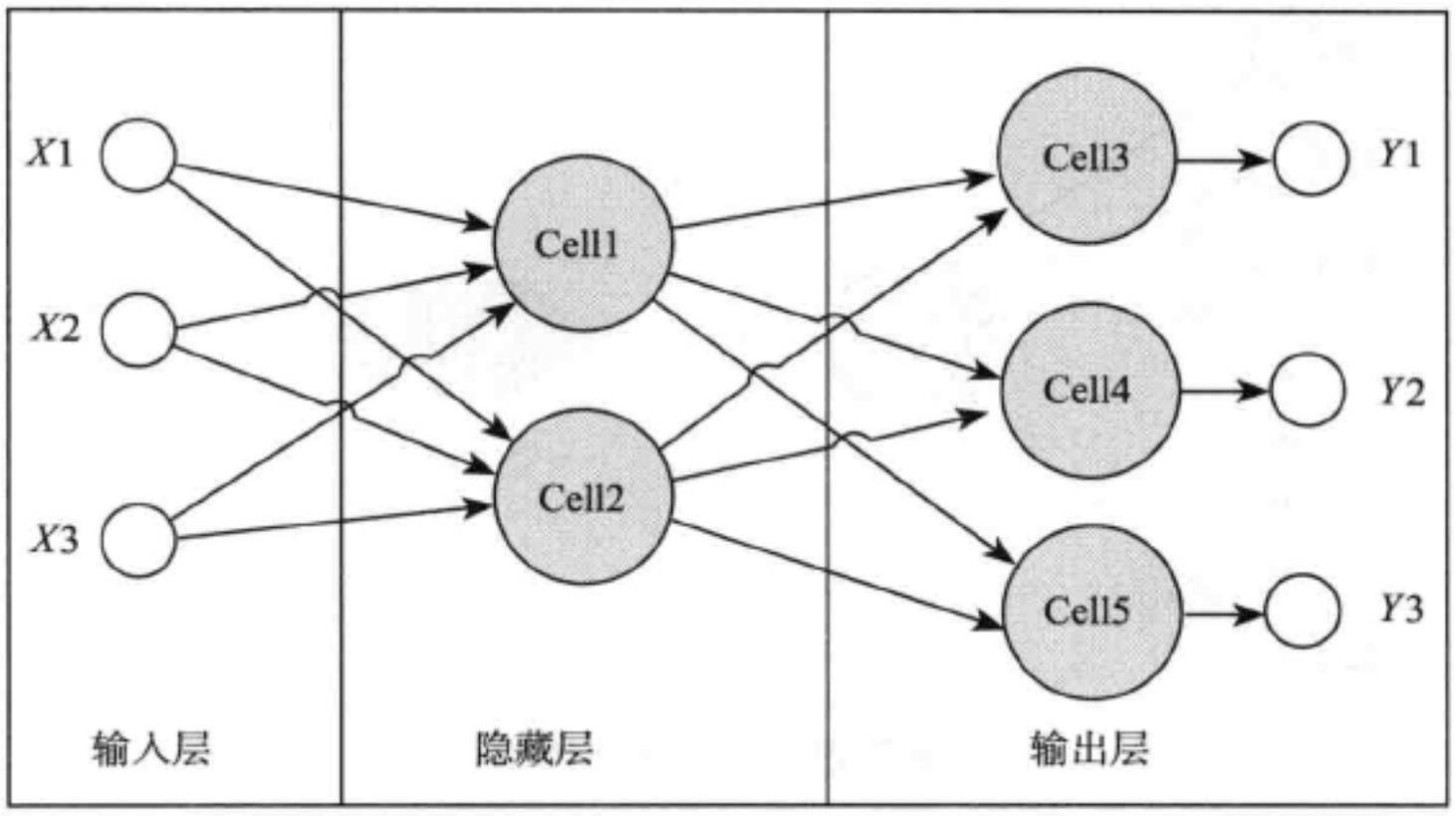


图 7-20　神经网络示意图

- 输入层：连接神经网络的所有输入变量，它并未做任何数据处理，只负责把每个输入变量与隐藏层中的神经元连接起来。
- 隐藏层：图 7-20 包含两个神经元，并把隐藏层中神经元的输出作为每个输出层中神经元的输入。一般来说两层网络（即一个隐藏层，一个输出层）就已经很充分了。但有时三层、四层或五层的网络结构（把隐藏层中的层级数逐步加大）会更有效。
- 输出层：对于二元分类分析问题来说，输出层往往只有一个神经元，输出结果为 0 还是 1 由输出值的取舍点分开。在多元分类问题中，输出层中神经元数的最大值为目标变量的取值分类数。在本例中，因为目标变量是三元分类型，其取值为 A、B、C，所以可以把原来的目标变量 *Y* 转换为 3 个二元变量：*Y*1（*Y* 等于 A，*Y* 不等于 A）和 *Y*2（*Y* 等于 B，*Y* 不等于 B），以及 *Y*3（*Y* 等于 C，*Y* 不等于 C）。最后使用 3 个神经元处理输出变量 *Y*1-3。对于回归分析，输出层只有一个神经元，这里就不详细介绍了。

7.4.3　反向传播算法

反向传播算法（backpropagetion）是神经网络中的经典算法，其基本原理如下。

1）设置神经网络的基本结构和初始参数，这里主要是设置：

- 隐藏层的层数和内部的神经元数目。
- 隐藏层和输出层内所有神经元的加权系数及偏差量。
- 所有神经元使用的激活函数。

2）把样本 1 中的自变量数据通过输入层正向输入至初始网络模型 ANN_{init} 中，并计算神经网络的输出 y_{out}（正向传播）。

3）计算样本 1 中目标变量与 y_{out} 的误差，并通过输出层反向输入初始网络模型中。通过计算误差的传播来调整各个神经元中加权系数和偏差量的取值（反向传播），最后得到由样本 1 修正后的神经网络模型 ANN_1。

4）重复上面两个步骤，逐次使用训练集中的样本数据修正神经网络模型中的参数，最后根据停止学习条件，停止模型的修改过程，并完成算法的建立。注意，训练集数据可以被重复学习，不必由于所有训练集均被遍历而停止学习。

1. 正向传播处理

这里以两层神经网络为例，其隐藏层的层级数为 1。在做正向传播处理时，首先遍历隐藏层的所有神经元，并计算其输出。其中 w.hidden 代表隐藏层中所有神经元内加权系数的矩阵（矩阵的行数等于神经元数，列数表示自变量数）。bias.hidden 代表神经元偏差量的向量（因为每个神经元内只有 1 个偏差量，所以隐藏层内所有神经元的偏差量使用向量表示即可）。ann.cell 是神经元的处理函数。相关函数代码如下：

```
o.hidden<-0
for(i in 1:nrow(w.hidden)){
    o.hidden[i]<-ann.cell(x,w.hidden[i,],bias.hidden[i])
}
```

遍历输出层的所有神经元，计算其输出 o.out，具体代码如代码清单 7-2 所示。

代码清单 7-2

```
o.out<-0
for(i in 1:nrow(w.out)){
    o.out[i]<-ann.cell(o.hidden,w.out[i,],bias.out[i])
}
```

其中 w.out 和 bias.out 分别表示输出层内神经元的加权系数和偏差量。

2. 反向传播处理

来看看反向传播处理时各层的表现。

输出层的误差 error.out：对于输出层的神经元 j，其误差可以计算为 $\text{Error}_j=O_j*(1-O_j)*(T_j-O_j)$。其中 O_j 为神经元 j 正向传播时计算的输出，T_j 是基于训练集样本中目标变量的实际取值。代码如下：

```
error.out<-o.out*(1-o.out)*(y-o.out)
```

其中，o.out 在代码清单 7-2 中计算，y 是样本的目标变量取值。

隐藏层的误差 error.hidden：对于隐藏层（这里以 1 层为例）的神经元 j，其误差可以计算为：

$$\text{Error}_j=O_j*(1-O_j)*\sum_k(\text{Error}_k*w_{jk})$$

其中，w_{jk} 为隐藏层内神经元 j 和与其相连的输出层内神经元 k 之间的加权系数值。Error_k 为神经元 k 的误差。代码如下：

```
error.hidden<-0
for(j in 1:nrow(w.hidden)){# 遍历所有的隐藏层内神经元
    error.tmp<-sum(error.out*w.out[,j])
```

```
    error.hidden[j]<-o.hidden[j]*(1-o.hidden[j])*error.tmp
}
```

下面使用神经元误差 error.out 和 error.hidden 对神经元内各加权系数 w 进行学习修正。

输出层神经元各加权系数的修正公式如下：

$$\Delta W_{jk}=l*\text{Error}_k*O_j\ ;\ W_{jk}=W_{jk}+\Delta W_{jk}。$$

其中 W_{jk} 为隐藏层内神经元 j 和与其相连的输出层内神经元 k 之间的加权系数值。l 为学习速率，其取值范围为 0 ～ 1。如果 l 取值过小，则学习的速度会非常缓慢；若 l 取值过大，则有可能造成学习模型不稳定。通常可以将 l 设置为 $1/t$，其中 t 为至本轮学习为止，所处理的整个训练集样本的循环次数。Error_k 为输出层中第 k 个神经元的误差 error.out[k]。O_j 是隐藏层中第 j 个神经元在做正向传播时的输出 o.hidden[j]。输图层的示意图如图 7-21 所示。

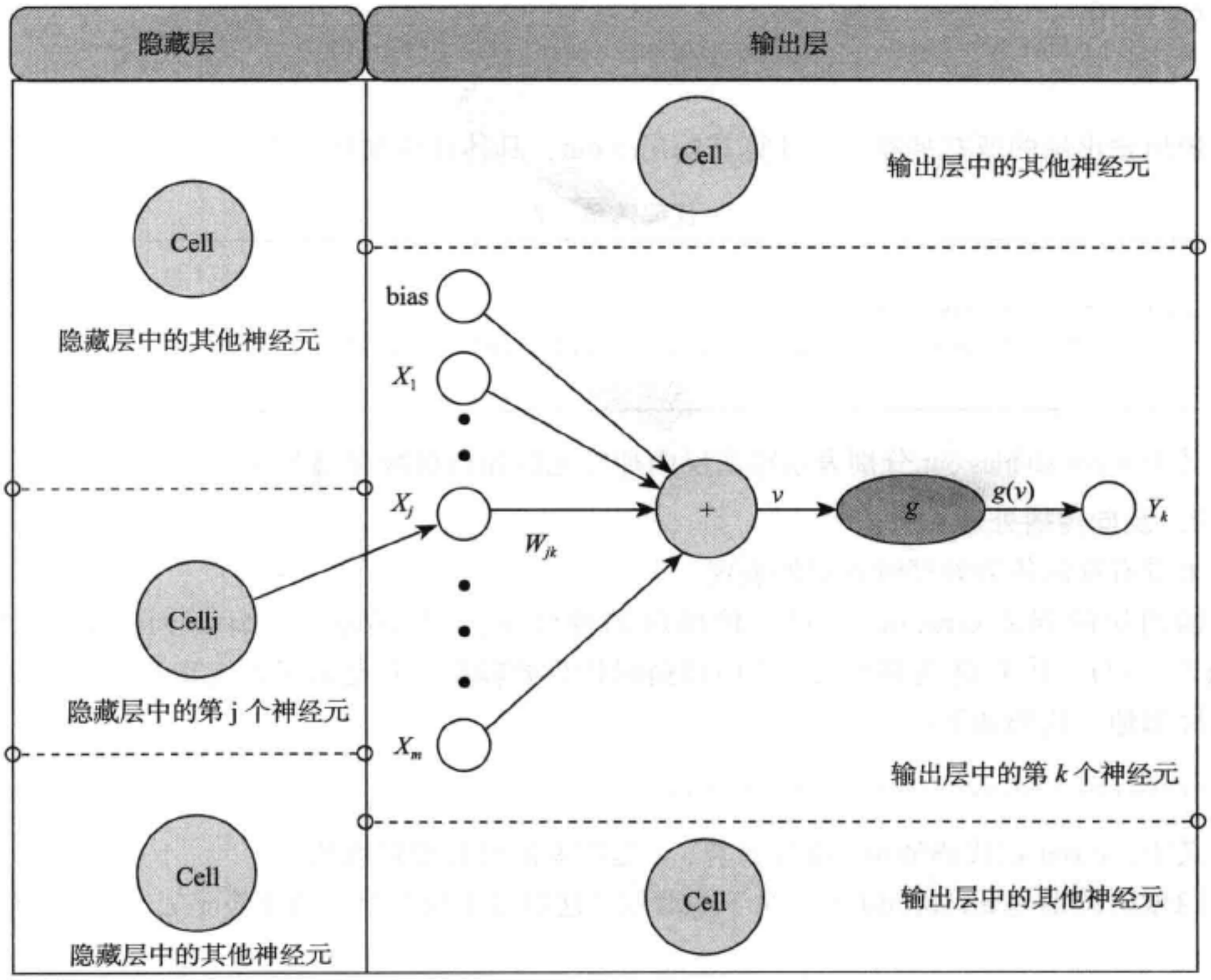


图 7-21 输出层示意图

遍历输出层内所有神经元的代码如下：

```
for(i in 1:length(y)){# 遍历所有的输出层内神经元
    w.out[i,]<-w.out[i,]+l*error.out[i]*o.hidden
}
```

隐藏层神经元各加权系数的修正公式和输出层类似，具体如下：

$$\Delta W_{jk}=1*\text{Error}_k*I_j \ ; \ W_{jk}= W_{jk} + \Delta W_{jk}。$$

其中 W_{jk} 为输入层内自变量 j 和与其相连的隐藏层内神经元 k 之间的加权系数值。Error_k 为隐藏层中第 k 个神经元的误差 error.hidden[k]。I_j 是第 j 个自变量 $x[j]$。隐藏层的示意图如图 7-22 所示。

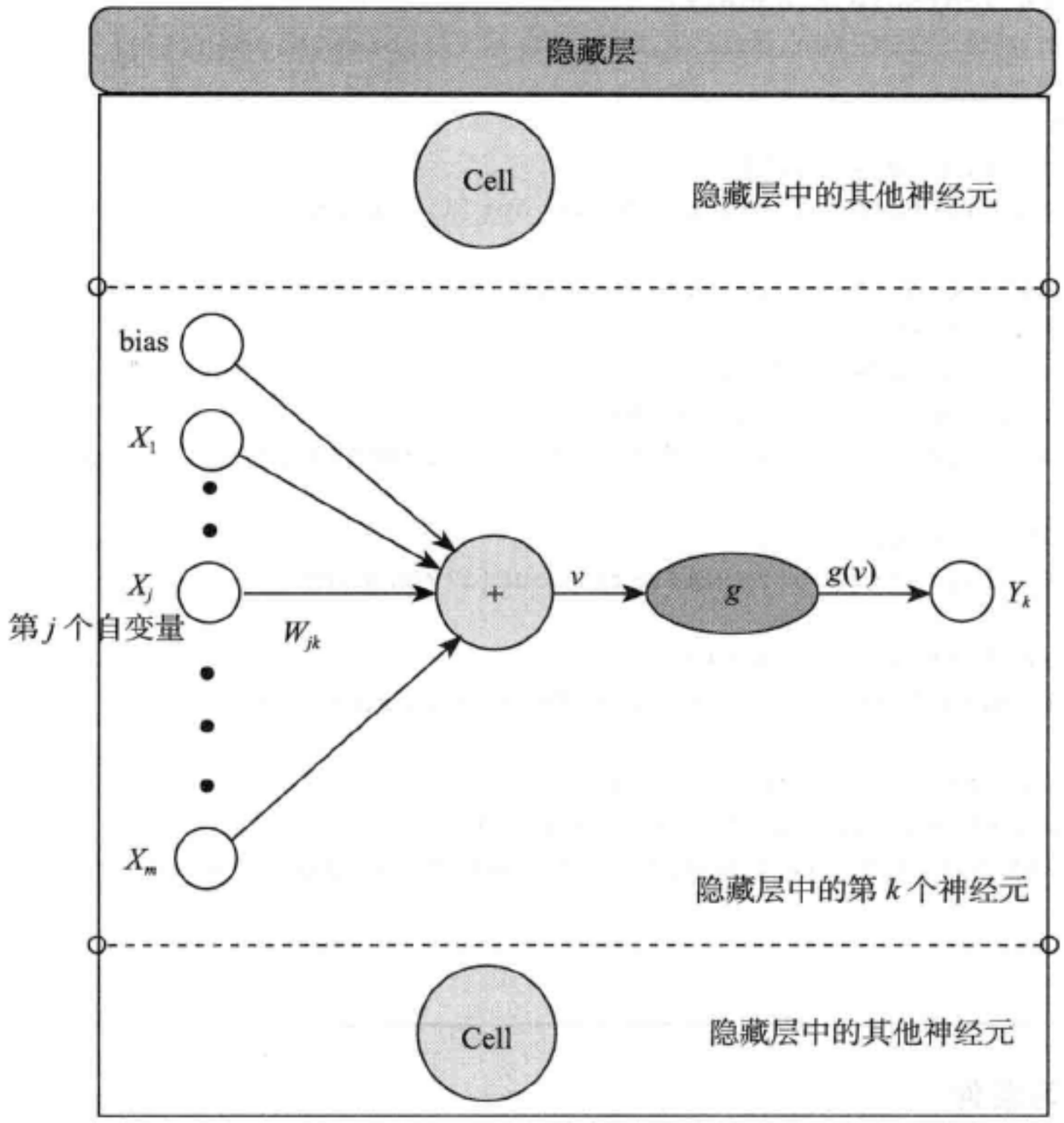


图 7-22 隐藏层示意图

遍历隐藏层内所有神经元的代码如下：

```
for(i in 1:nrow(w.hidden)){# 遍历所有的隐藏层内神经元
    w.hidden[i,]<-w.hidden[i,]+1*error.hidden[i]*x
}
```

下面使用神经元误差 error.out 和 error.hidden 对神经元内各偏差量 bias 进行学习修正，公式如下：

$$\Delta \text{bias}_k=1*\text{Error}_k \ ; \ \text{bias}_k = \text{bias}_k + \Delta \text{bias}_k。$$

其中 bias_k 是神经元的偏差值，Error_k 是每个神经元的误差。相关代码如下：

```
bias.out<-bias.out+1*error.out
bias.hidden<-bias.hidden+1*error.hidden
```

最后综合本节中的各个代码，组成用于一次模型学习的函数 ann.core。该函数的具体代码如代码清单 7-3 所示。

代码清单　7-3

```
ann.core<-function(x,y,w.hidden,bias.hidden,w.out,bias.out,l){
    o.hidden<-0
    for(i in 1:nrow(w.hidden)){
        o.hidden[i]<-ann.cell(x,w.hidden[i,],bias.hidden[i])
    }
    o.out<-0
    for(i in 1:nrow(w.out)){
        o.out[i]<-ann.cell(o.hidden,w.out[i,],bias.out[i])
    }
    error.out<-o.out*(1-o.out)*(y-o.out)
    error.hidden<-0
    for(j in 1:nrow(w.hidden)){
        error.tmp<-sum(error.out*w.out[,j])
        error.hidden[j]<-o.hidden[j]*(1-o.hidden[j])*error.tmp
    }
    for(i in 1:length(y)){
        w.out[i,]<-w.out[i,]+l*error.out[i]*o.hidden
    }
    for(i in 1:nrow(w.hidden)){
        w.hidden[i,]<-w.hidden[i,]+l*error.hidden[i]*x
    }
    bias.out<-bias.out+l*error.out
    bias.hidden<-bias.hidden+l*error.hidden
    out<-list(w.hidden=w.hidden,bias.hidden=bias.hidden,w.out=w.out,bias.out=bias.
    out)
    out
}
```

3. 停止学习条件

停止学习的条件一般分为如下 3 种情况（任意一个条件成立，则停止学习）。

1）每次使用一个样本进行学习时，所有神经元内的加权系数差 ΔW_{jk} 均小于指定的阈值。

2）被错误分类的样本占总样本数的比例小于指定的阈值。

3）执行了指定次数的处理循环。

下面的代码清单 7-4 以指定循环次数为停止条件，建立两层神经网络自定义函数。

代码清单　7-4

```
#x 和 y 是样本输入，必须全是矩阵，每行是一个样本
#hidden.num 为隐藏层的神经元数
#learn.num 为学习的循环次数上限
ann<-function(x,y,hidden.num,learn.num){
    #### 初始化各个神经元内的加权系数和偏差值，随机地在 -0.5 ~ 0.5 中选取 ####
    init<-seq(length.out=100,from=-0.5,to=0.5)
```

```
w.hidden<-matrix(sample(init,ncol(x)*hidden.num),nrow=hidden.num)
bias.hidden<-sample(init,hidden.num)
w.out<-matrix(sample(init,hidden.num*ncol(y)),nrow=ncol(y))
bias.out<-sample(init,ncol(y))
t<-((1:learn.num)-1)%/%nrow(x)+1
k<-((1:learn.num)-1)%%nrow(x)+1
#### 使用训练集样本进行 learn.num 次学习 ####
for(i in 1:learn.num){
    l<-1/t[i]# 本次学习速率
    j<-k[i]
   ann.new<-ann.core(as.numeric(x[j,]),as.numeric(y[j,]),
            w.hidden,bias.hidden,w.out,bias.out,l)
    w.hidden<-ann.new$w.hidden
    bias.hidden<-ann.new$bias.hidden
    w.out<-ann.new$w.out
    bias.out<-ann.new$bias.out
  }
  out<-list(w.hidden=w.hidden,bias.hidden=bias.hidden,w.out=w.out,bias.
  out=bias.out)
  out
}
```

下面来检验代码清单 7-1、代码清单 7-3 以及代码清单 7-4 建立的模型算法的效果。首先建立用于模型预测的函数。代码如下：

```
ann.priedict.core<-function(x,sol.ann){
    o.hidden<-0
    for(i in 1:nrow(sol.ann$w.hidden)){# 遍历每个隐藏层的 cell
        o.hidden[i]<-ann.cell(x,sol.ann$w.hidden[i,],sol.ann$bias.hidden[i])
    }
    o.out<-0
    for(i in 1:nrow(sol.ann$w.out)){
        o.out[i]<-ann.cell(o.hidden,sol.ann$w.out[i,],sol.ann$bias.out[i])
    }
    o.out<-as.numeric(o.out>0.5)
    o.out
}
ann.priedict<-function(x,sol.ann){
    y<-matrix(0,nrow=nrow(x),ncol=length(sol.ann$bias.out))
    for(i in 1:nrow(x)){
        y[i,]<-ann.priedict.core(x[i,],sol.ann)
    }
    y
}
```

ann.priedict 是预测函数，其中输入参数 sol.ann 为由 ann 函数产生的模型参数对象。*X* 为样本数据的自变量。下面使用 R 语言自带的 iris 数据集做实例。代码如下：

```
x=iris[,1:4]
y1=as.numeric(iris$Species=="setosa")
```

```
y2=as.numeric(iris$Species=="versicolor")
y3=as.numeric(iris$Species=="virginica")
y<-matrix(c(y1,y2,y3),ncol=3)
ann.sol<-ann(x,y,25,20000)
ann.priedict(x,ann.sol)
```

其中，iris$Species 为目标变量，取值为 "setosa" 和 "versicolor" 以及 "virginica"，记为 y1-y3。这里通过 ann 函数建立一个神经网络模型 ann.sol，并使用 ann.priedict 函数完成目标变量的预测工作。

7.4.4 R 语言实现

7.4.3 节自定义了 ann 函数来实现一层隐藏层神经网络模型。在 R 语言中，nnet 包的 nnet 函数也可以建立一层隐藏层神经网络模型，对任意闭区间的连续函数均可以使用这样的一种神经网络来逼近实际结果。而对于分类分析，因为其目标变量并非是连续型变量，但是目标变量取 1 的概率却是连续型变量，所以我们去逼近目标变量取 1 的概率。nnet 函数的参数如表 7-17 所示。

表 7-17 nnet 函数的参数

参数	说明
size	隐藏层中的神经元数，设置为 0 时，表示建立一层神经网络
Wts	初始系数，如果不设定，则使用随机数设定
linout	如果等于 TRUE，则模型的输出为连续型实数，一般用于回归分析（目标变量为连续型）；如果等于 FALSE（默认取值），则模型输出为逻辑数据，一般用于分类分析（目标变量为分类型）。在进行分类分析时，也可以把 linout 设为 TRUE，再添加一个阶跃函数转化为逻辑型输出
maxit	最大迭代次数 iterations，默认为 100 次
abstol 和 reltol	均是停止迭代学习的标准，默认值均为 1×10^{-4}
skip	是否跳过隐藏层。如果为 FALSE（默认），则不跳过
decay	加权系数的衰减

另外，formula、data、subset、na.action、weights 参数可以参考 7.1.3 节中 rpart 函数的说明。下面使用 7.1.2 节中的数据集进行人工神经网络分析。代码如下：

```
library(nnet)
nnet.sol<-nnet(mod.formula,data=data.train,size=10,maxit=1000)
    # weights:  171
    initial  value 3788.708030
    iter  10 value 3556.456959
          <中间省略>
    iter 180 value 3187.311981
    iter 190 value 3187.288179
    final  value 3187.015041
    converged
```

这里输入层的神经元数为 15，输出层的神经元数是 1，隐藏层的神经元数设置为 10

(将 size 参数设置为 10)。最大迭代次数 maxit 设置为 1000，即每个训练集样本最多要学习 1000 次。由结果可知，迭代 190 次就达到了 abstol 和 reltol 参数的标准，从而停止了神经网络的学习。一般，尽量将 maxit 设置得大一些，然后观察结果 final value 上显示的迭代次数，并且最好保证它小于 maxit 的设置值。

summary(nnet.sol) 可以输出神经元的加权系数，如图 7-23 所示。

```
> summary(nnet.sol)
a 15-10-1 network with 171 weights
options were - entropy fitting
   b->h1   i1->h1   i2->h1   i3->h1   i4->h1   i5->h1   i6->h1   i7->h1   i8->h1   i9->h1  i10->h1  i11->h1  i12->h1  i13->h1  i14->h1  i15->h1
 -716.55  1394.88    -4.96  1313.16 -1716.02   -71.06  1965.64  2481.74  1292.52   -30.97  1786.46 -2080.73  -939.80 -1585.03   919.55 -1011.18
   b->h2   i1->h2   i2->h2   i3->h2   i4->h2   i5->h2   i6->h2   i7->h2   i8->h2   i9->h2  i10->h2  i11->h2  i12->h2  i13->h2  i14->h2  i15->h2
   -0.67     0.31    -8.49    -1.58     0.22    -3.02    -0.11    -0.57    -0.29    -0.64    -0.67    -0.22     0.46     0.13     0.31    -0.60
   b->h3   i1->h3   i2->h3   i3->h3   i4->h3   i5->h3   i6->h3   i7->h3   i8->h3   i9->h3  i10->h3  i11->h3  i12->h3  i13->h3  i14->h3  i15->h3
  124.18  -557.46   -13.92   762.96   409.20  -790.40   996.41   -49.31  -355.52  1504.17  2155.32   453.59 -1174.66   760.03     3.02  -820.53
   b->h4   i1->h4   i2->h4   i3->h4   i4->h4   i5->h4   i6->h4   i7->h4   i8->h4   i9->h4  i10->h4  i11->h4  i12->h4  i13->h4  i14->h4  i15->h4
    0.17     0.69     1.85     2.66     0.78     3.07     0.82     0.30    -0.56    -0.12    -0.19    -0.58     0.06    -0.40    -0.18    -0.32
   b->h5   i1->h5   i2->h5   i3->h5   i4->h5   i5->h5   i6->h5   i7->h5   i8->h5   i9->h5  i10->h5  i11->h5  i12->h5  i13->h5  i14->h5  i15->h5
  510.44  1550.16  -129.65   -28.57  -769.81   593.19    95.77  -210.79  2923.62 -1275.08 -1116.58    -5.14  -900.61  -473.56  2406.25   143.24
   b->h6   i1->h6   i2->h6   i3->h6   i4->h6   i5->h6   i6->h6   i7->h6   i8->h6   i9->h6  i10->h6  i11->h6  i12->h6  i13->h6  i14->h6  i15->h6
   -0.18     0.57    -0.70    -0.58     0.19    -0.50    -0.36    -0.30    -0.53    -0.63    -0.42     0.36     0.51    -0.24     0.04     0.11
   b->h7   i1->h7   i2->h7   i3->h7   i4->h7   i5->h7   i6->h7   i7->h7   i8->h7   i9->h7  i10->h7  i11->h7  i12->h7  i13->h7  i14->h7  i15->h7
   -0.66     0.51    -1.53    -0.85    -0.32    -0.84    -0.50    -0.14    -0.11    -0.36    -0.68    -0.28     0.50    -0.31     0.59    -0.09
   b->h8   i1->h8   i2->h8   i3->h8   i4->h8   i5->h8   i6->h8   i7->h8   i8->h8   i9->h8  i10->h8  i11->h8  i12->h8  i13->h8  i14->h8  i15->h8
-2012.41 -2513.28   154.06   146.25  -348.40  -173.69  -256.80   329.86  -908.82 -2243.32 -1632.78   837.06 -1803.40  2121.68     0.68     1.89
   b->h9   i1->h9   i2->h9   i3->h9   i4->h9   i5->h9   i6->h9   i7->h9   i8->h9   i9->h9  i10->h9  i11->h9  i12->h9  i13->h9  i14->h9  i15->h9
 -757.17  -511.47   235.85  -568.35  -574.54   230.12   155.33     3.85  -646.77   384.01  -172.45  1523.97     8.72   125.80     0.56    -0.62
  b->h10  i1->h10  i2->h10  i3->h10  i4->h10  i5->h10  i6->h10  i7->h10  i8->h10  i9->h10 i10->h10 i11->h10 i12->h10 i13->h10 i14->h10 i15->h10
  -21.89    28.37   177.67   332.92   -29.91   299.57    19.26     2.32    -7.88     4.02     4.20    89.92     2.38     2.62    -0.15     0.15
    b->o    h1->o    h2->o    h3->o    h4->o    h5->o    h6->o    h7->o    h8->o    h9->o   h10->o
 -305.55    -1.18     0.21     1.99  -305.61    -0.92     0.11    -0.52  3026.46 -2096.92  -317.43
```

图 7-23 模型结果输出

在图 7-23 的显示结果中，前 10 行依次是 b,i1,i2,..,i15->h1,h2,..,h10 的权重系数，即隐藏层中 10 个神经单元的系数。最后一行为 b,h1,h2,..,h10->o 的权重系数，即输出层中 1 个神经单元的系数。

使用 predict 函数同样可以在模型 nnet.sol 的基础上预测新的数据。代码如下：

```
test.pred.prob<-predict(nnet.sol,data.test)
test.pred.class<-ifelse(test.pred.prob>0.5,1,0)
test.pred.class<-as.numeric(predict(nnet.sol,data.test,type="class"))
```

这里的 predict(nnet.sol,data.test) 可以理解为 nnet.sol 模型预测目标变量取 1 的概率，其值处于 [0,1] 的范围内。使用 ifelse(test.pred.prob>0.5,1,0) 来设置一个阈值 0.5，如果 test.pred.prob 大于 0.5，则 test.pred.class 取 1，小于 0.5 则取 0。也可以直接把 predict 函数中的 type 参数设置为 class 来实现。

7.4.5 隐藏层中神经单元数目的确定

在 nnet 中可使用参数 size 来设置隐藏层中神经元的个数，其确定准则包括如下一些方式（这里只给出了包含一层隐藏层的神经网络情况）。

方式 1 如果输出层数目较少，则：

- size 一定处于 $[3,2n_i+1]$ 范围内，其中 n_i 是输入层的单元数。
- 依据 n_i 来确定 size（clementine 中的多层训练法），具体如表 7-18 所示。

表 7-18　n_i 和 size 对照表

输入层数 n_i	1	2	3	3+k
隐藏层数 size	3	4	7	7+2k-1

当 n_i 大于 3 后每增加 1，size 对应增加 2k-1。

快速确定法：size=λ*n_i，其中系数 λ 的取值范围为 1.2 ～ 1.5 。

方式 2 如果输出层数目较多，则：

❑ Size=max(3,(n_i+n_o)/20)，其中 n_o 是输出层中的神经单元数。

❑ Size=a+(n_i+n_o)$^{1/2}$，其中 a 的取值范围为 1 ～ 10。

注意，以上均为一些经验值，至今还没有一套完美的理论来计算如何设置 size 参数。我们只能依次尝试，并比较分类器指标的大小。例如，针对上例，可以尝试把 size 设置为 30（由方式 1 得到）或 18、20、21、23（由方式 2 得到）。

7.5　分类器的性能评估

7.5.1　混淆矩阵

1. 基本概念

混淆矩阵（Confusion Matrix）用于把样本实际值（true class）和模型预测值（predicted class）进行联列表分析。如果实际样本 true class 取 −1，则为反例（negative），取 +1，则为正例（positive）。如果模型预测 predicted class 错误，则为假（false），预测正确，则为真（true）。最后混淆矩阵如表 7-19 所示。

表 7-19　混淆矩阵

		True class	
		−1	+1
Predicted class	−1	True negative（TN 真反例）	False negative（FN 假反例）
	+1	False positive（FP 假正例）	True positive（TP 真正例）
合计		N=TN+FP	P=FN+TP
		C=N+P	

表 7-19 中各个数据的含义如表 7-20 所示。

表 7-20　表 7-19 中各数据的含义

	含义
TP	模型预测为正 positive（预测为 1），并且预测正确 true（实际样本值也是正）的样本数
TN	模型预测为负 negative（预测为 −1），并且预测正确 true（实际样本值也是负）的样本数
FP	模型预测为正 positive（预测为 1），而预测错误 false（实际样本值却为负）的样本数
FN	模型预测为负 negative（预测为 −1），而预测错误 false（实际样本值却为正）的样本数
N	表示实际样本 True class 中为 −1 的样本数，N=TN+FP

（续）

	含义
P	表示实际样本 True class 中为 +1 的样本数，P=FN+TP
C	表示所有样本数，C=N+P

这里以 7.4 节中的神经网络分析为例来进行说明，实现混淆矩阵的代码如下：

```
library(nnet)
nnet.sol<-nnet(mod.formula,data=data.train,size=30,maxit=1000)
con.matrix<-table(data.frame(predicted=predict(nnet.sol,data.
test)>0.5,actual=data.test$HanDynastyArt==1)); con.matrix
                actual
predicted       FALSE   TRUE
  FALSE           743     298
  TRUE            349     807
```

首先以 30 个神经单元为隐藏层，从而得到模型 nnet.sol，并使用 predict 函数来预测测试集。这里使用阈值 0.5 为标准，如果模型预测目标变量取 1 的概率大于 0.5，则取 TRUE，如果小于 0.5，则取 FALSE。由于原始样本的目标变量 data.test$HanDynastyArt 的值为 factor0 和 1，因此，把等于 1 的转化为 TRUE，等于 0 的转化为 FALSE。最后进行联列表分析。

对应于 TP、TN、FP、FN 等概念，再引入相应的比率概念，如表 7-21 所示。

表 7-21 TNR、FNR、FPR、TPR 指标含义

		True class	
		−1	1
Predicted class	−1	True negative rate（TNR 真反率） = TN/(TN+FP)=TN/N	False negative rate（FNR 假反率） =FN/(FN+TP)=FN/P
	1	False positive rate（FPR 假正率） = FP/(TN+FP)=FP/N	True positive rate（TPR 真正率） =TP/(FN+TP)=TP/P

同样，这里的 TNR 也称为特指度 Specicity，TPR 称为灵敏度 Sensitivity 和召回率（或查全率）Recall，FPR 称为 Fallout，FNR 称为漏查率 Miss，其具体的意义和应用会在后续的章节中讲解。

2. 正确率和错误率

正确率是指模型预测正确的概率 $P(Y_{预测}=Y_{真实})$，其公式如下：

$$\text{Accuracy} = (TP+TN)/C$$

同样，错误率是指模型预测错误的概率 $P(Y_{预测} \neq Y_{真实})$，其公式如下：

$$\text{Error rate} = (FP+FN)/C$$

在混淆矩阵示例中 nnet.sol 模型的正确率和错误率依次为：

```
acc<- sum(con.matrix[c(1,4)])/sum(con.matrix) ;acc
[1] 0.7055075
```

```
err<- sum(con.matrix[c(2,3)])/sum(con.matrix) ;err
[1] 0.2944925
```

可见模型的正确预测率可以达到70%以上。不过需要注意的是，由于在划分训练集和测试集数据前，针对分类不均衡的现象做了处理，即对目标变量值为1的数据进行重复采样，导致data.test中很多样本数据是重复的，因此这里的正确率和错误率并非完全精确，一般情况下，正确率会略高一些。

3. 查准率和查全率

在实际应用中，对目标变量取正例和负例的关注程度并不相同。例如，在研究用户是否购买新书《西汉艺术简史》的示例中通常更加关心模型对变量值为1（表示购买）时预测的精度，此时把目标变量值为1（表示购买）设为正例，同时引入查准率和查全率这两个概念来衡量分类器对正例样本预测的准确性和可信性。

查准率（precision）：正确分类的正例数（TP）占模型预测正例（TP+FP）的比例，即：

Precision=TP/(TP+FP)

该指标用于衡量分类器预测正例的准确性。

查全率（recall）：正确分类的正例数（TP）占实际正例（P=TP+FN）的比例，即：

Recall=TPR=TP/(TP+FN)

该指标用于衡量分类器预测正例的可信性。例如，模型sol的查准率可以达到90%，即在所有预测为正例的样本中，有90%符合实际情况。但是查全率低，如只有10%时，即该分类器只能预测出实际情况中10%的正例，也就是说，这样的分类器仍然无用。

注意

查全率也叫灵敏度和召回率。

对于上文得到的nnet.sol模型，其查准率和查全率依次为：

```
precision<- con.matrix[4]/sum(con.matrix[c(2,4)]) ; precision
[1] 0.6980969
recall<- con.matrix[4]/sum(con.matrix[c(3,4)]); recall
[1] 0.7303167
```

即模型中预测为正例的准确性为69.8%，并可以预测出实际情况中73%的正例。

另外，可以使用F1指标来综合考虑查准率和查全率，其定义为查准率和查全率的调和平均数：

F1=2*Precision*Recall/(Precision+Recall)

对于上文得到的nnet.sol模型，其F1指标为：

```
2*precision*recall/(precision+recall)
[1] 0.7138434
```

7.5.2 ROC 曲线和 AUC

除了使用查准率和查全率以外，针对只关注正例的分类器指标，更常用的是受试者特征曲线 ROC 和曲线下面积 AUC。

1. 灵敏度和特指度

灵敏度（sensitivity）：正确分类的正例数（TP）占实际正例（P=TP+FN）的比例，即：

$$Sensitivity=TPR=TP/(TP+FN)$$

特指度（specicity）：正确分类的反例数（TN）占实际反例（N=TN+FP）的比例，即：

$$Specicity=TNR=TN/(TN+FP)$$

$$1\text{-}Specicity=FPR=FP/(TN+FP)$$

2. 受试者特征曲线 ROC

（1）ROC 的定义

分类器可以预测出目标变量取正例时的概率。以 nnet.sol 为例：

```
test.pred.prob<-predict(nnet.sol,data.test)
```

上面返回的就是概率数值。如果设置一个阈值为 0.5，大于 0.5 的预测为正例，小于 0.5 的预测为反例，那么就可以达到常规的分类预测值 test.pred.class。代码如下：

```
test.pred.class<-ifelse(test.pred.prob>0.5,1,0)
```

从而可以得到一个混淆矩阵，以及一对灵敏度和特指度的值。

如果变换阈值，让阈值依次为 0、0.001、0.002、…0.999、1，则可以得到 1001 组灵敏度和特指度的值，然后依次把灵敏度和 1- 特指度绘制于平面中，即可得到 ROC 曲线。绘制代码如下：

```
test.pred.prob<-predict(nnet.sol,data.test)
sens<-c();spec<-c()
for(i in 0:1000){
   threshold.i<-i/1000# 阈值取 0,0.001,0.002,..,0.999,1
con.matrix<-table(data.frame(predicted=test.pred.prob>threshold.i,actual=data.
test$HanDynastyArt==1))
   sens[i+1]<- con.matrix[4]/sum(con.matrix[c(3,4)])
   spec[i+1]<- con.matrix[1]/sum(con.matrix[c(1,2)])
}
plot(1- spec,sens ,type="l", main="ROC 图 ",ylab=" 真正率 (TPR)= 灵敏度 (Sensitivity)",
xlab=" 假正率 (FPR)=1- 特指度 (1-Specicity)")
```

绘制结果如图 7-24 所示。

其中，ROC 曲线经过点（1,1），即特指度 =0，灵敏度 =1，表示所有样本均被预测为正例。如果经过点（1,0），即特指度 =0，灵敏度 =0，则表示所有样本均被预测为负例。

（2）何时使用 ROC 曲线来衡量分类器

ROC 曲线的横轴是假正率（FPR），它等于 1- 特指度（1-Specicity），纵轴是真正率

(TPR)，它等于灵敏度（sensitivity）。使用 ROC 曲线来衡量所考虑的目的是：在尽量少的“误诊”（假正率 FPR）基础上，尽可能多地检验出正例的个体（真正率 TPR）。

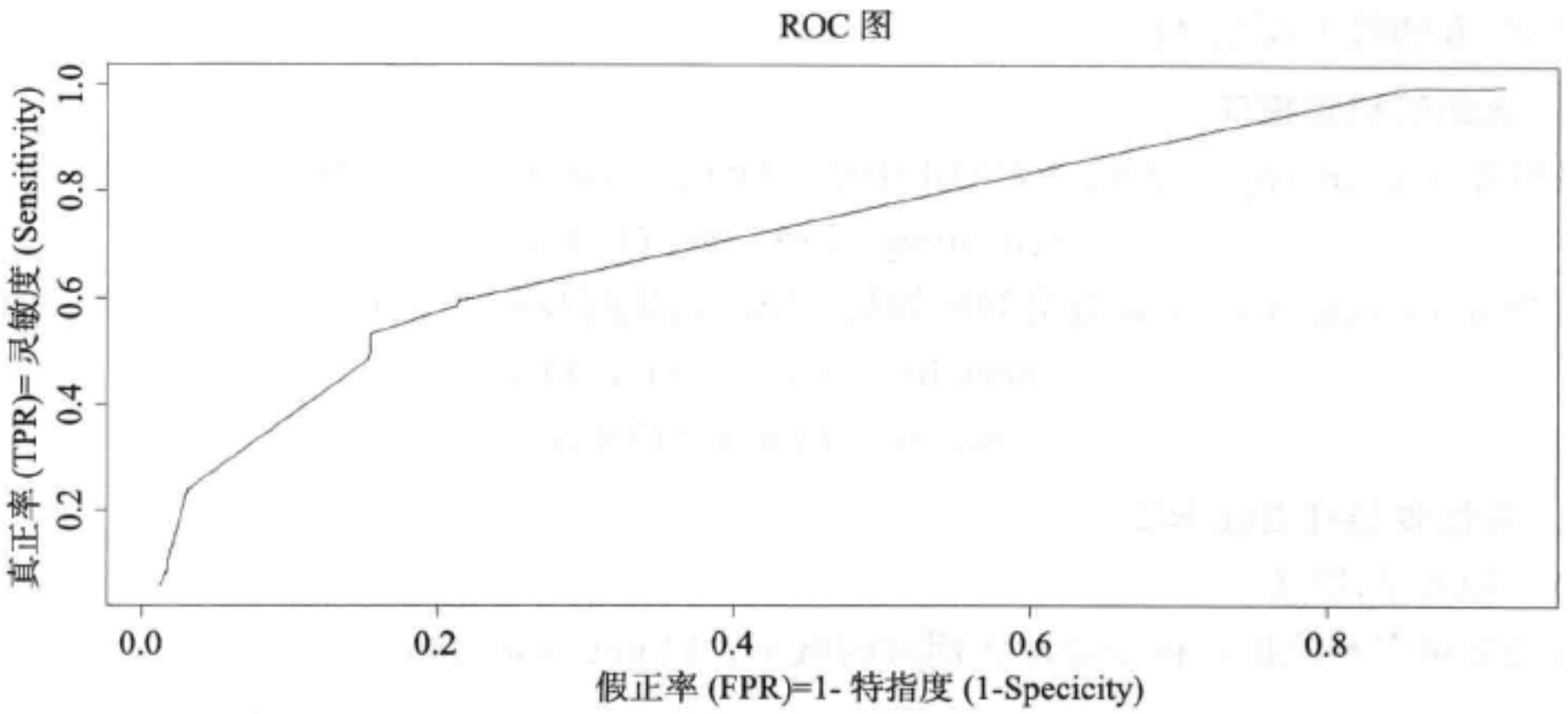


图 7-24　ROC 图

例如，想通过某登录行为的部分变量 *x*1、*x*2、…*xn* 来预测该次登录行为是否为机器人登录，则用 *y* 来表示（等于 1 表示是机器人登录，等于 0 则表示不是）。可以通过如下两个指标来综合考虑分类器 *y*=*f*(*x*1, *x*2, ..*xn*) 的性能。

- 由于我们更加关心“是机器人登录”的样本，所以希望分类器可以尽可能多地检测出“是机器人登录”的样本，即指标真正率 TPR 尽量大。
- 如果把实际上“不是机器人登录”的样本误判为“是机器人登录”，则会导致真实用户有不好的体验（因为在检测到机器人登录后，系统通常会返回相应的警告提示），所以希望分类器可以尽可能少的误判，即指标假正率 FPR 要尽量小。

此时即可使用 ROC 曲线来综合衡量分类器。

3. ROCR 包的应用

以上指标数据均是使用 R 语言的最基本函数实现的，实际上用 ROCR 包可便捷地实现上述以及后续提到的众多指标。其步骤如下：首先加载 ROCR 包，然后使用 prediction 函数对数据格式进行转化，得到一个可被 ROCR 包的核心函数 performance 使用的数据对象，最后使用 performance 函数计算各种数据指标，并用 plot 函数绘制相关曲线。

这里仍使用 nnet.sol 模型来讲解 ROCR 包的使用。代码如下：

```
library(nnet)
nnet.sol<-nnet(mod.formula,data=data.train,size=30,maxit=1000)
test.real<-as.numeric(ifelse(data.test$HanDynastyArt==0,0,1))
test.pred.class<-as.numeric(predict(nnet.sol,data.test,type="class"))
predictions<-prediction(test.pred.class,test.real)
```

注意，prediction 函数的第一个参数是模型预测的值，第二个参数是测试集的实际样本

值。对象 test.pred.class 和 test.real 均是 0 和 1 的数值型向量（其中正例设为 1，反例设为 0），代码如下：

```
is(test.real)
[1] "numeric" "vector"
is(test.pred.class)
[1] "numeric" "vector"
```

然后使用 performance 函数计算灵敏度。代码如下：

```
performance(predictions, measure='sens')@y.values
[[1]]
[1] 0.0000000 0.7303167 1.0000000
```

由于 prediction 函数的第一个参数是模型预测出的分类数值（只能取 0 和 1），所以计算出的灵敏度只有 3 个数值，其中第二个为实际的灵敏度。

```
performance(predictions, measure='sens')@y.values[[1]][2]
[1] 0.7303167
```

同样把参数 measure 设置为 spec 得到特指度，代码如下：

```
performance(predictions,'spec')@y.values[[1]][2]
[1] 0.6804029
```

如果使用模型预测为正例的概率数值（取值在 0 ～ 1 的连续型数值）作为 prediction 函数的第一个参数，则计算的灵敏度和特指度是包括 151 个点的向量，即阈值取 [0，1] 范围内的 151 个值时，对应的灵敏度和特指度数值为：

```
test.pred.prob<-predict(nnet.sol,data.test)
predictions<-prediction(test.pred.prob,test.real)
sens<-performance(predictions, measure='sens')@y.values[[1]]
spec<-performance(predictions,'spec')@y.values[[1]]
```

最后用下面的语句绘制 ROC 曲线。

```
plot(1- spec,sens ,type="l", main="ROC 图 ",ylab=" 真正率 (TPR)= 灵敏度 (Sensitivity)",
xlab=" 假正率 (FPR)=1- 特指度 (1-Specicity)")
```

绘制结果如图 7-25 所示。

也可以使用 ROCR 包中的 plot 函数来绘制。代码如下：

```
plot(performance(predictions,'tpr','fpr'), main="ROC 图 ",ylab=" 真正率 (TPR)= 灵敏度
(Sensitivity)",xlab=" 假正率 (FPR)=1- 特指度 (1-Specicity)",colorize=T)
```

绘制结果如图 7-26 所示。

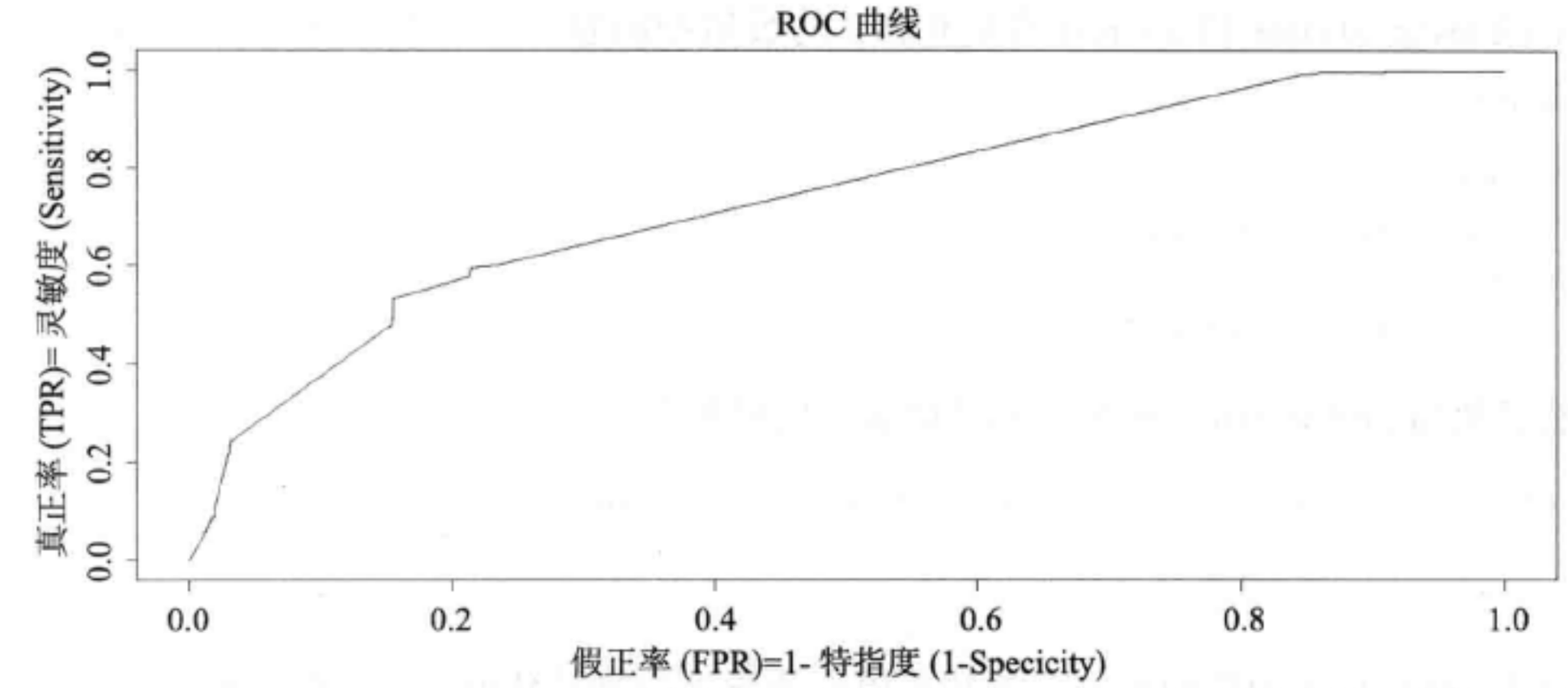


图 7-25　ROC 曲线

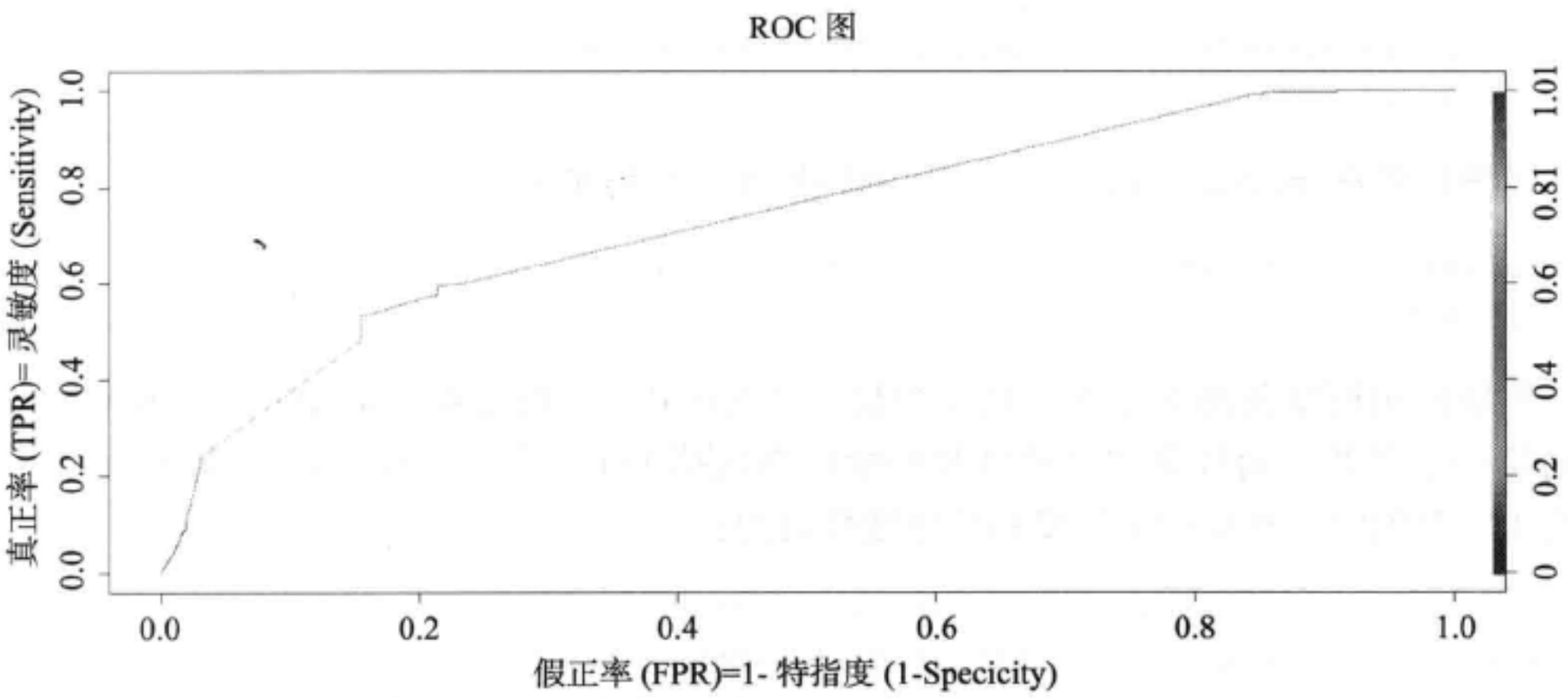


图 7-26　ROC 图

4. 曲线下面积 AUC

如果 ROC 曲线经过点（0,1），即特指度 =1，灵敏度 =1，则表示为最优的分类器，然而绝大多数的 ROC 曲线并非如此，此时可通过引入 ROC 曲线下的面积 AUC 来衡量不同模型间 ROC 曲线的表现情况。AUC 的面积越大，该模型的 ROC 曲线表现越好，模型越可用。

在 ROCR 包中，把 performance 函数中的 measure 参数设置为 auc 可以得到 AUC 数值。代码如下：

```
predictions<-prediction(test.pred.prob,test.real)
performance(predictions, measure='auc')@y.values[[1]]
[1] 0.7340307
```

这里笔者补充了一个针对 svm、决策树、神经网络等多个模型的 ROC 曲线分析代码，可以通过 http://www.bassary.com/?p=1536 下载。

7.5.3 提升度和提升曲线

1. 提升度的定义和含义

提升度 lift 的定义是：lift=TPR/((TP+FP)/C)，即提升度 lift 等于 TPR 除以预测正例的比例，这里把 (TP+FP)/C 称为深度 depth。例如：若 10000 个消费者中有 1000 个是潜在的客户，则向这 10000 个消费者发放促销短信的效率是 10%（即 1000 个客户会通过促销短信浏览网站，即响应率是 10%）。通过研究这 10000 个消费者，建立分类模型进行分类，从而得到有可能比较积极的 1000 个消费者（通过预测为正例的概率进行排序，并取前 1000 个样本）。如果此时这 1000 个消费者中有 300 个是潜在客户，则命中率为 30%=TPR。此时的提升值 lift=30%/10%=3，客户的响应率提升至原先的 3 倍，提高了投入产出比。

在 ROCR 包中，把 performance 函数中的 measure 参数设置为 lift，可以得到提升度数值。代码如下：

```
predictions<-prediction(test.pred.prob,test.real)
lift<-performance(predictions, measure='lift')@y.values[[1]]
```

把参数 measure 设置为 rpp 可以得到深度数值。代码如下：

```
depth<-performance(predictions, measure='rpp')@y.values[[1]]
```

2. 提升曲线

提升曲线是以深度为横轴，以提升度为纵轴绘制的曲线。绘制代码如下：

```
plot(depth,lift ,type="l", main="lift 图 ",ylab=" 提升度 (lift)",xlab=" 深度 (depth)")
```

绘制结果如图 7-27 所示。

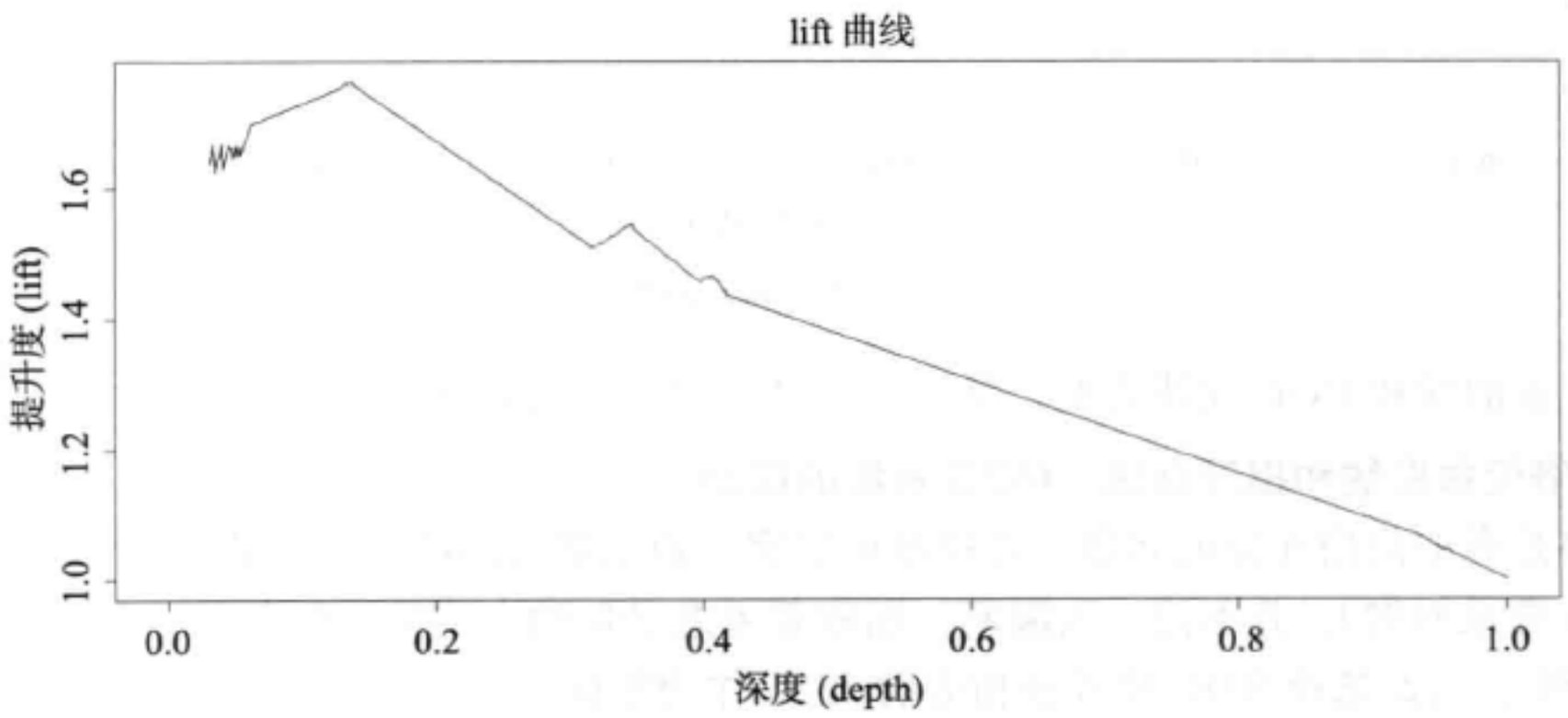


图 7-27 提升曲线

当阈值为 0 时，所有的样本都被预测为正例，因此 depth=1，TPR=1，于是 lift=1，模型未起提升作用。随着阈值逐渐增大，被预测为正例的样本数逐渐减少，depth 减小，而较少的预测正例样本中的真实正例之比例逐渐增大。当阈值增大至 1 时，没有样本被预测为正例，

此时 depth=0，TPR=0，于是 lift=0/0=NA。由图 7-27 可见，lift 与 depth 存在向相反方向变化的关系。与 ROC 曲线不同，lift 曲线凸向（0，1）点。一个好的分类模型会在尽量大的 depth 下得到尽量大的 lift（当然要大于 1），也就是说该模型 lift 曲线的右半部分应该尽量陡峭。

7.5.4　洛伦兹曲线

1. 洛伦兹曲线定义

洛伦兹（lorenz）曲线和提升曲线类似，只是将提升曲线的纵轴由 lift(=TPR/depth) 改为了 TPR，也就是说，该曲线的纵轴是 TPR，横轴是深度 depth=(TP+FP)/C。绘制洛伦兹曲线的代码如下：

```
predictions<-prediction(test.pred.prob,test.real)
tpr<-performance(predictions, measure='tpr')@y.values[[1]]
depth<-performance(predictions, measure='rpp')@y.values[[1]]
plot(depth,tpr ,type="l", main="lorenz 图 ",ylab=" 查全率 (tpr)",xlab=" 深度 (depth)")
```

绘制结果如图 7-28 所示。

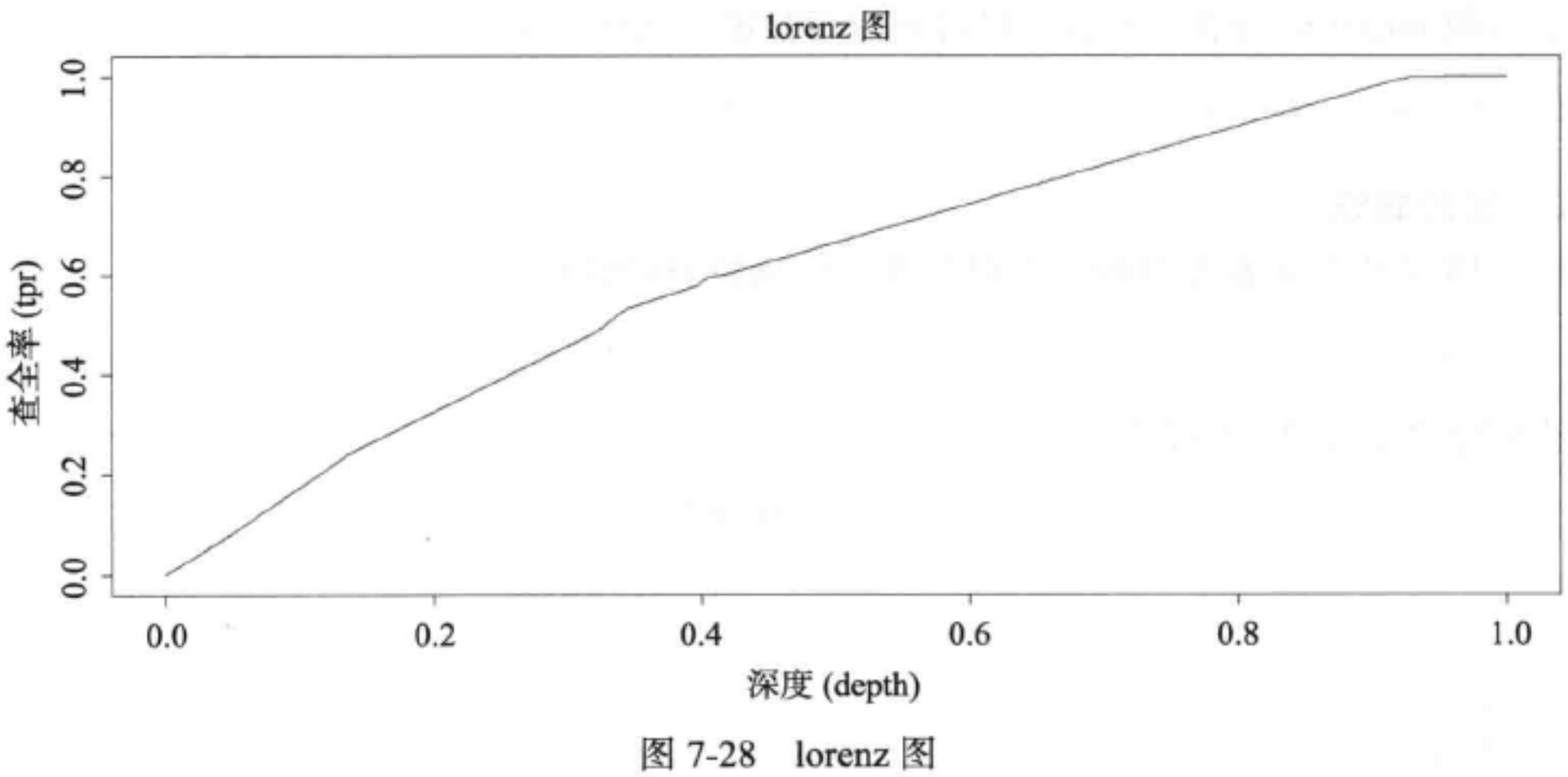


图 7-28　lorenz 图

洛伦兹曲线和 ROC 曲线类似，是一个凸向（0,1）点的曲线。

2. 洛伦兹曲线和提升曲线、ROC 曲线的区别

如果是类似信用评分的问题，希望尽可能完全地识别出那些有风险的客户（如预测是否为机器人登录问题），并不使一人漏网，则需要考虑尽量增大 TPR（查全率），同时减小 FPR（减少误判），因此选择 ROC 曲线及相应的 AUC 作为指标。

如果是类似精确营销的问题，希望通过对全体消费者进行分类，从而得到具有较高响应率的客户群（如预测用户购买新书问题），以便提高投入产出比，则需要考虑尽量提高 lift(提升度)，同时 depth 不能太小（如果只给一个消费者发放宣传信息，虽然响应率较大，却无法得到足够多的响应），因此选择提升曲线或洛伦兹曲线作为指标。

第8章
样本细分

从本章开始，讲解研究样本的相关知识，主要包括样本的细分（第8章），样本间的关联和网络关系（第9章）。

对于分析师而言，对样本进行细分是一项重要工作——要在复杂的原始数据中去除大量无关的冗余数据，进而提取出主要信息，并把样本进行归类。实际上对于网站分析和问卷调查分析来说，只有把类似的样本（用户）分开来研究才有意义，其指标数据才更有代表性（只代表该特定群体样本）。

本章首先讲解如何对变量进行降维（因子分析），即使用最少的变量来代表最多的原始信息。接着介绍样本的归类（kmeans聚类分析），最后介绍knn判别分析法，并以此来构建一套基于商品的推荐系统。

8.1 数据降维

数据降维主要有主成分分析和因子分析两种方式。可以认为主成分分析是一种特殊情况下的因子分析。本节重点讲解使用相对更广的因子分析。

8.1.1 问题引入

假设有 n 个指标变量 $X_1,X_2,\cdots,X_{n-1},X_n$，其中每个变量包括 k 个样本，记为 X_{kn}（组成一个 $k\times n$ 的矩阵），如表8-1所示，其中，id是样本序列号 $1\sim k$，每列为 $1\sim n$ 个指标变量。

表8-1 数据集

id	X_1	X_2	…	X_{n-1}	X_n
1	x_{11}	x_{12}	…	x_{1n-1}	x_{1n}
2	x_{21}	x_{22}	…	x_{2n-1}	x_{2n}
…	…	…	…	…	…
k	x_{k1}	x_{k2}	…	x_{kn-1}	x_{kn}

这 n 个指标相互之间可能有很强的相关性，如果 $X1$、$X5$ 和 $X8$ 之间的相关性较大，就可以使用一个新变量 $F1$ 综合代表 $X1$、$X5$、$X8$ 这 3 个变量，实现数据的降维，我们称新变量 $F1$ 为一个因子。因子分析实际就是计算出 m 个因子 F，使其符合如下关系式。

$$F_{km} * A_{mn}+e_{kn}=X_{kn} \tag{8-1}$$

由公式 8-1 可知，因子 F 是把原始变量 X 中的公共信息提取出来形成的。另外，因子数 m 明显小于原始变量数 n，自然也就实现了数据的降维。值得注意的是，提取因子后，AF 不能完全代表 X（还要加误差 e），因为信息量有所损失。降维后，因子数目 m 越小，信息损失量越大，所以这时要平衡数据降维程度和信息量的损失。在因子分析中，A 为因子载荷，e 为特殊因子，详见 8.1.2 节。

8.1.2 因子分析概述

1. 因子载荷

因子载荷（loading）是公式 8-1 中的 A，其定义式为：

$$A_{mn}=\mathrm{Cov}(X_{kn}, F_{km})$$

即在因子载荷 A 中，元素 a_{ij} 表示第 i 列原始变量与第 j 列因子的相关系数 $\mathrm{Cov}(X_i,F_j)$。其值越接近 ±1，降维后的因子与原始变量的相关性越强。对因子载荷的估算分为主成分分析（princomp）、主因子分析（factor）、极大似然法（likelihood）等三种方法。其中，极大似然法是主流的方法。它们的数学原理这里就不一一讲解了。

2. 因子的含义和因子载荷的旋转

可以通过因子载荷来推测出因子的实际含义。例如，研究如表 8-2 所示的某网站对于目标用户细分的问卷调查数据。

表 8-2 问卷调查问题

	问题	单选项
1	只要衣服的款式和设计喜欢，我愿意付更多的钱	1 分至 10 分
2	网上买衣服，我喜欢选择专门经营服装的电商品牌	1 分至 10 分
3	如果买到一次质量差的衣服，我再也不会买这个品牌的衣服	1 分至 10 分
4	在网上购物我最看重的就是客户服务（包括送货 / 退换货政策等）	1 分至 10 分
5	网上买衣服，价格低很重要	1 分至 10 分
6	我认为买衣服没有必要多花钱	1 分至 10 分
7	服务好是我选择到网站的主要原因	1 分至 10 分
8	价格低是网站最吸引我的地方	1 分至 10 分
9	我平时比较关注衣服的款式设计和流行趋势	1 分至 10 分
10	同类服装我会选择价格最低的	1 分至 10 分
11	买衣服时我首先考虑的是衣服的质量和做工	1 分至 10 分
12	衣服的款式和设计是我买衣服最看重的因素	1 分至 10 分

针对这 12 个问题，用户可以通过选择 1 ～ 10 分来表示其认同程度，1 分代表“很不同

意”，10 分代表“非常赞同”。 这里把用户回答的 12 道题目的得分（变量为 $X1 \sim X12$）降维到 4 个公共因子（$F1 \sim F4$），并把得到的因子载荷数据绘制成如图 8-1 所示的条形图，如果 a_{ij} 绝对值大于 0.6（标明有较大的相关性）则标红。

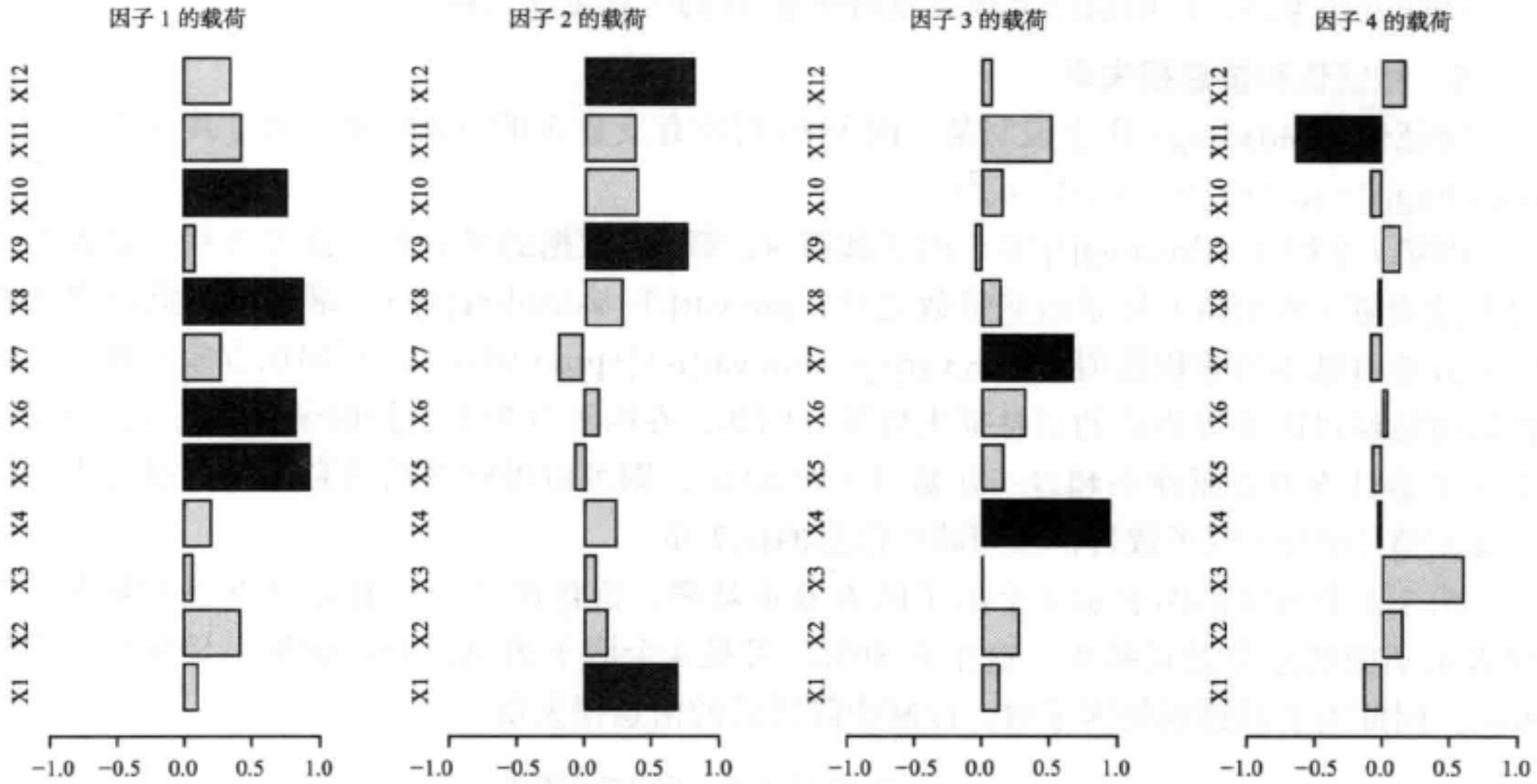


图 8-1 因子载荷示意图

其中因子 1 与问题 5、6、8、10 的相关性最强，其问题的内容依次是：

- 网上买衣服，价格低很重要。
- 我认为买衣服没有必要多花钱。
- 价格低是网站最吸引我的地方。
- 同类服装我会选择价格最低的。

上述 4 个问题均和价格有关系，此时可认为因子 1 能够很好地衡量用户对商品价格因素的考虑，进而完成数据降维。

如果因子载荷每列的内部元素 a_{ij} 间的方差较小，则很难确定因子的含义，此时一般会对因子载荷做方差最大的正交旋转，使每列内元素 a_{ij} 的平方向 0 和 1 两极分化，从而更好地区分某一因子代表的是哪一种原始变量。

3. 特殊因子

特殊因子是公式 8-1 中的 e，其定义式为：

$$e_{kn}=X_{kn}-F_{km}*A_{mn}$$

即经过数据降维后信息的损失量。

4. 对原始变量方差的解释

对于某一原始变量 Xi，其数据的方差可以拆解为公因子方差（由所有因子 F 引起的方差）和特殊方差（uniquenesses)(由所有特殊因子 e 引起的方差)：

Xi 变量方差 = 公因子方差 + 特殊方差

如果把数据进行标准化处理，即每个变量 *Xi* 的方差为 1，则此时：

1= 公因子方差 + 特殊方差

特殊方差越高，说明数据降维后原始变量 *Xi* 的信息量损失越大。

5. 特征值和信息损失率

特征值（ssloading）用于说明某一因子 *Fi* 对所有变量 *X* 的方差贡献，其计算式为：

$\text{ssloading}[i]= (a_{i1}^2+a_{i2}^2+..+a_{in}\text{-}1^2+a_{in}^2)$

即第 *i* 个因子 ssloading[*i*] 等于因子载荷 A_{mn} 第 *i* 列数据的平方和。方差贡献率是各因子的方差贡献（特征值）与原始变量数之比：pro.var[*i*]= ssloading[*i*]/n。累积方差贡献率是各因子方差贡献率的累积数值：cum.var[*i*]= cum.var[*i*-1]+pro.var[*i*]。通过累积方差贡献率可以整体衡量经过因子分析后的信息损失情况。例如，在因子分析前，把因子数设为 4，如果各个因子累计方差贡献率不超过一定数目（如 80%），则可以说降维后信息的损失量过大，此时要调整（增加）因子数目，从而减少信息的损失量。

图 8-2 中的柱形图表示 4 个因子的方差贡献率，蓝色曲线表示其累计方差贡献率，红线表示期望的总方差贡献率，这里是 80%。可见 4 个因子的总方差贡献率不足我们希望的 80%，因此需要继续添加因子数，以减少降维后的信息损失量。

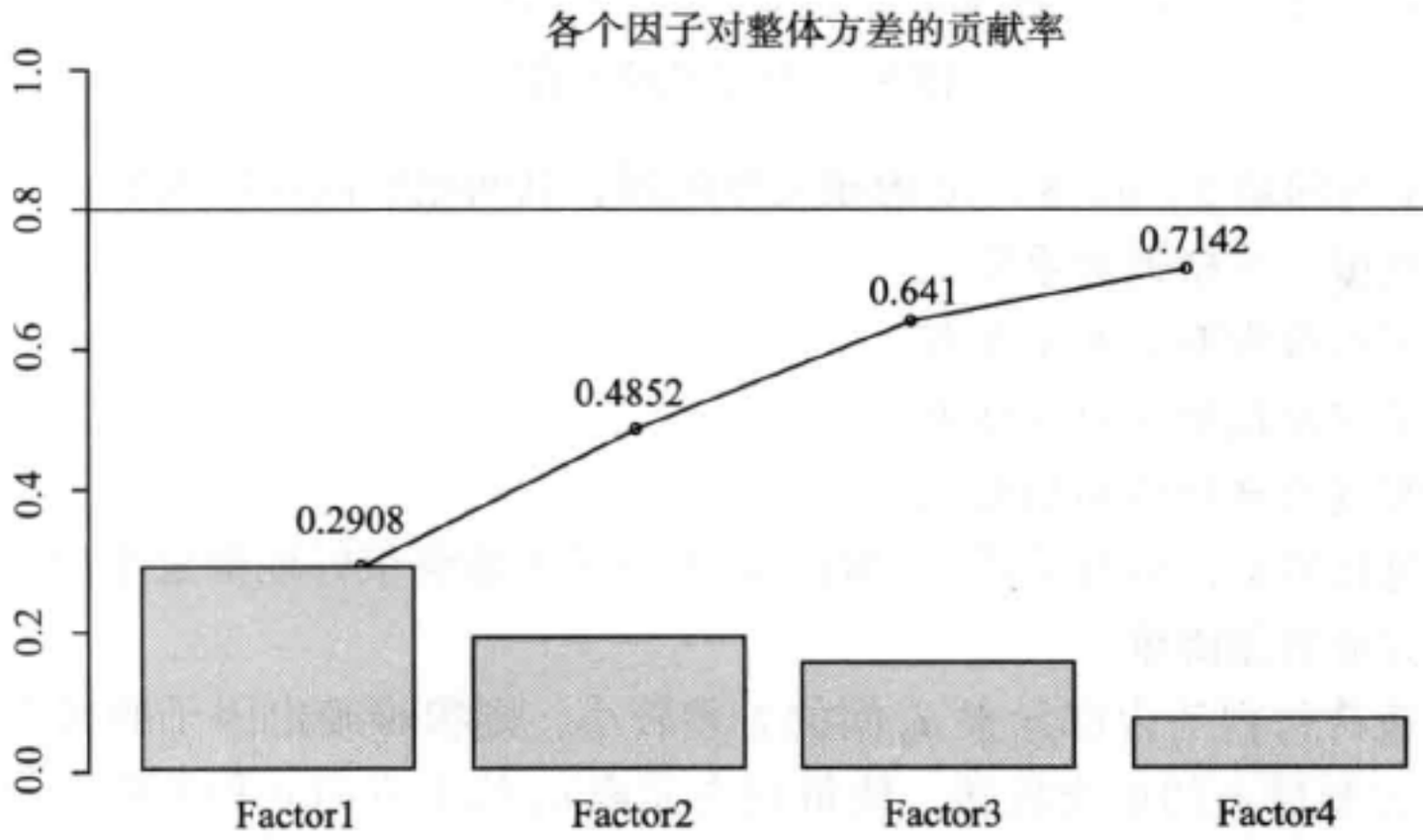


图 8-2 方差累计贡献率示意图

6. 因子得分

在计算好因子载荷 *A* 和特殊因子 *e* 后，就可以计算因子 *F* 的数据了。这里有加权最小二乘法（Bartlett）和回归法（regression）两种计算方式，其具体数据公式就不一一讲解了。

8.1.3 factanal 函数

在 R 语言中，factanal 函数采用主流的最大似然法来估算因子载荷 A，从而实现因子分

析过程。其参数说明如表 8-3 所示。

表 8-3 factanal 函数参数说明

参数	说明
x 和 data	指明原始变量数据，相当于公式 8-1 中的 X_{kn} 如果 data 设置为数据框，则 x 可以作为 formula 指明使用了 data 中的哪些列变量 如果不设置 data，则可以把数据 X_{kn} 直接以 kxn 阶矩阵的形式赋予 x
factors	因子数，相当于公式 8-1 中的 m
subset	用于过滤样本数据，即过滤 data 或 x 数据的行数据
rotation	设置因子载荷的旋转 如果设置为“varimax”，则使用方差最大的正交旋转 如果设置为“none”，则不旋转
scores	计算因子得分的方法 如果设置为“none”，则计算因子得分 如果设置为“regression”，则以回归法计算因子得分 如果设置为“Bartlett”，则以加权最小二乘法计算因子得分

下面结合一个网站问卷调查数据的实例来说明 factanal 函数的使用以及进行因子分析的一般步骤。

8.1.4 实例：问卷调查的因子分析

下面对表 8-2 的问卷调查数据进行因子分析。我们知道，因子分析的主要目的是从这 12 个问题数据（1 ～ 12 个变量 X）中提取出可以代表用户购买行为的主要信息。

1. 读入数据

这里收集了 48 份问卷数据，以 csv 格式存储于文件中，具体如表 8-4 所示。

表 8-4 数据集

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}
1	6	7	2	5	8	7	8	8	3	8	7	10
2	9	10	5	8	10	9	9	10	5	9	8	10
…	…	…	…	…	…	…	…	…	…	…	…	…
48	7	8	3	6	9	8	9	7	4	9	8	10

读取 csv 数据的代码如下：

```
root<-"C:/Users/liming/Desktop/写书/chap8/lib/"
data<-read.csv(paste(root,"applicant.csv",sep=""))
data<-data[-1]
```

产生的数据框如下，这里展示数据框 data 的前两条数据。

```
data[,1:2]
  X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12
1  6  7  2  5  8  7  8  8  3   8   7  10
2  9 10  5  8 10  9  9 10  5   9   8  10
```

2. 原始变量间的相关系数

首先分析原始变量间的相关性，使用 cor 函数得到变量 $X1 \sim X12$ 间的相关性系数矩阵。

```
cor(data)
```

	X1	X2	X3	X4	X5	X6
X1	1.00000000	0.2388057	0.044040889	0.306313037	0.092144656	0.22843205
X2	0.23880573	1.0000000	0.123419296	0.379614151	0.430769427	0.37125894
X3	0.04404089	0.1234193	1.000000000	0.001589766	0.001106763	0.07682449
X4	0.30631304	0.3796142	0.001589766	1.000000000	0.302439887	0.48277493
X5	0.09214466	0.4307694	0.001106763	0.302439887	1.000000000	0.80754502
X6	0.22843205	0.3712589	0.076824494	0.482774928	0.807545017	1.00000000
X7	-0.10674947	0.3536910	-0.030269601	0.645408595	0.410090809	0.35584446
X8	0.27069919	0.4895490	0.054727421	0.361643880	0.799630538	0.81802080
X9	0.54837963	0.1409249	0.265585352	0.140723415	0.015125832	0.14720197
X10	0.34557633	0.3405493	0.093522030	0.393164148	0.704340067	0.69751518
X11	0.46720619	0.2840928	-0.323319352	0.685155768	0.482455962	0.52683562
X12	0.58591822	0.3842084	0.140017368	0.326957419	0.250283416	0.41614467

	X7	X8	X9	X10	X11	X12
X1	-0.106749472	0.27069919	0.54837963	0.34557633	0.4672062	0.585918216
X2	0.353690969	0.48954902	0.14092491	0.34054927	0.2840928	0.384208365
X3	-0.030269601	0.05472742	0.26558535	0.09352203	-0.3233194	0.140017368
X4	0.645408595	0.36164388	0.14072342	0.39316415	0.6851558	0.326957419
X5	0.410090809	0.79963054	0.01512583	0.70434007	0.4824560	0.250283416
X6	0.355844464	0.81802080	0.14720197	0.69751518	0.5268356	0.416144671
X7	1.000000000	0.23990754	-0.15593849	0.28018499	0.4482455	0.002755617
X8	0.239907539	1.00000000	0.25541758	0.81473421	0.5632842	0.558035847
X9	-0.155938495	0.25541758	1.00000000	0.33722821	0.2149532	0.692636173
X10	0.280184989	0.81473421	0.33722821	1.00000000	0.6128077	0.622554062
X11	0.448245522	0.56328419	0.21495316	0.61280767	1.0000000	0.395798842
X12	0.002755617	0.55803585	0.69263617	0.62255406	0.3957988	1.000000000

这里把矩阵中绝对值大于 0.6 的数据都加上底纹了，可见很多变量间存在相关性，即可以使用因子分析来对其降维。

3. 使用最大似然法建立因子分析模型

（1）模型建立

这里使用 8.1.3 节中的 factanal 函数进行因子分析建模。代码如下：

```
sol<-factanal( ~ .,data=data,factors=4,scores= "Bartlett",rotation= "varimax")
```

设置 scores 参数为“ Bartlett”，即使用加权最小的二乘法做因子得分计算（计算最后的因子数据），并使用 sol$score 直接读取。设置 rotation 参数为“ varimax”，即对因子载荷进行方差最大的正交旋转。

（2）显著性检验

结果模型 sol 中有显著性检验的 pvalue 数据：sol$PVAL，如果该数据小于典型显著度数值 0.05 或无法得到显著性检验的 pvalue 数据，则表明未通过显著性检验。此时要适当提高因子（factanal 函数的 factors 参数）数。代码如下：

```
if(is.null(sol$PVAL)|(sol$PVAL<0.05)){
    print("结果未通过检验! ")
    return(sol)
}
```

4. 对各原始变量的方差解释

在得到因子模型 sol 的结果后，可以查看特殊方差（Uniquenesses），即可由特殊因子 e 解释的变量 $X1 \sim X12$ 的方差，或者直接理解为信息损失。查看到的情况如下：

```
sol$uniquenesses
       X1        X2        X3        X4        X5        X6        X7
0.4685517 0.6900097 0.6325198 0.0050000 0.1205627 0.2002187 0.4361284
       X8        X9       X10       X11       X12
0.1227568 0.3731419 0.2223482 0.0050000 0.1532241
```

由结果可知，因子分析后，$X2$、$X3$ 变量的信息损失量最大，已经大于 0.5 了，即因子 F 对其的方差解释小于特殊因子 e 的方差解释。

5. 因子含义的解释

下面通过 sol$loadings 来查看因子载荷 A，其返回值是一个 12×4 的矩阵。

```
sol$loadings
Loadings:
      Factor1 Factor2 Factor3 Factor4
X1            0.697   0.120  -0.150
X2    0.414   0.186   0.277   0.165
X3                            0.596
X4    0.190   0.242   0.948
X5    0.920           0.150
X6    0.827   0.125   0.313
X7    0.278  -0.181   0.668
X8    0.881   0.289   0.130
X9            0.774           0.136
X10   0.761   0.410   0.154
X11   0.423   0.386   0.514  -0.635
X12   0.349   0.831           0.171
< 以下不展示 >
```

在矩阵中，元素 a_{ij} 是变量 Xi 和因子 Fj 对应的相关系数，可依据绝对值大于 0.55 的 a_{ij} 以及变量 $X1 \sim X12$ 对应的问卷问题来分析因子与哪些问题相关，最后找出因子的含义，具体见表 8-5。

表 8-5 因子的含义

因子	因子对应的问卷问题	因子含义
Factor1	$X5$(0.92): 网上买衣服，价格低很重要 $X6$(0.827): 我认为买衣服没有必要多花钱 $X8$(0.881): 价格低是网站最吸引我的地方 $X10$(0.761): 同类服装我会选择价格最低的	商品价格因素

（续）

因子	因子对应的问卷问题	因子含义
Factor2	*X*1(0.697): 只要衣服的款式和设计喜欢，我愿意付更多的钱 *X*9(0.774): 我平时比较关注衣服的款式设计和流行趋势 *X*12(0.831): 衣服的款式和设计是我买衣服最看重的因素	商品样式因素
Factor3	*X*4(0.948): 在网上购物我最看重的就是客户服务（包括送货 / 退换货政策等） *X*7(0.668): 服务好是我选择到网站的主要原因	网站服务因素
Factor4	*X*11(-0.635): 买衣服时我首先考虑的是衣服的质量和做工 *X*3(0.596): 如果买到一次质量差的衣服，我再也不会买这个品牌的衣服	商品质量因素

此外，变量 *X*2 不能归为任何因子中的解释。把上面数据绘制成一组条形图可更好地查看结果。绘制代码如下：

```
loadings.abs.std=0.55
factors=4
tmp.x=0
   par(mfrow=c(1,factors))
    for(i in 1: factors){
        tmp.x.factor<-0
        for(j in 1:length(names(data))){
            if(abs(loadings(sol)[,i][j])>loadings.abs.std){
                tmp.x<-c(tmp.x,j)
                tmp.x.factor<-c(tmp.x.factor,j)
            }
        }
        data1<-sol$loadings[,i]
        data1[-tmp.x.factor]<-0
        data2<-sol$loadings[,i]
        data2[tmp.x.factor]<-0
        barplot(data1,horiz=TRUE,main=paste("因子",i,"的载荷"),col="red",xlim=
        c(-1,1))
        barplot(data2,horiz=TRUE,add=TRUE)
    }
```

这里把 a_{ij} 大于 0.55 的元素置为红色，结果如图 8-3 所示。

6. 各因子对原始变量整体的方差解释

同样，仍可以通过 sol$loadings 来查看方差贡献（特征值）（SS loadings）、方差贡献率（Proportion Var），以及累积方差贡献率（Cumulative Var）。代码如下：

```
sol$loadings
       <以上不展示>
                Factor1 Factor2 Factor3 Factor4
   SS loadings      3.489   2.334   1.869   0.879
   Proportion Var   0.291   0.194   0.156   0.073
   Cumulative Var   0.291   0.485   0.641   0.714
```

在这里，factanal 函数并未给出直接读取以上数据的方法，但是可以通过因子载荷间接

计算出来。

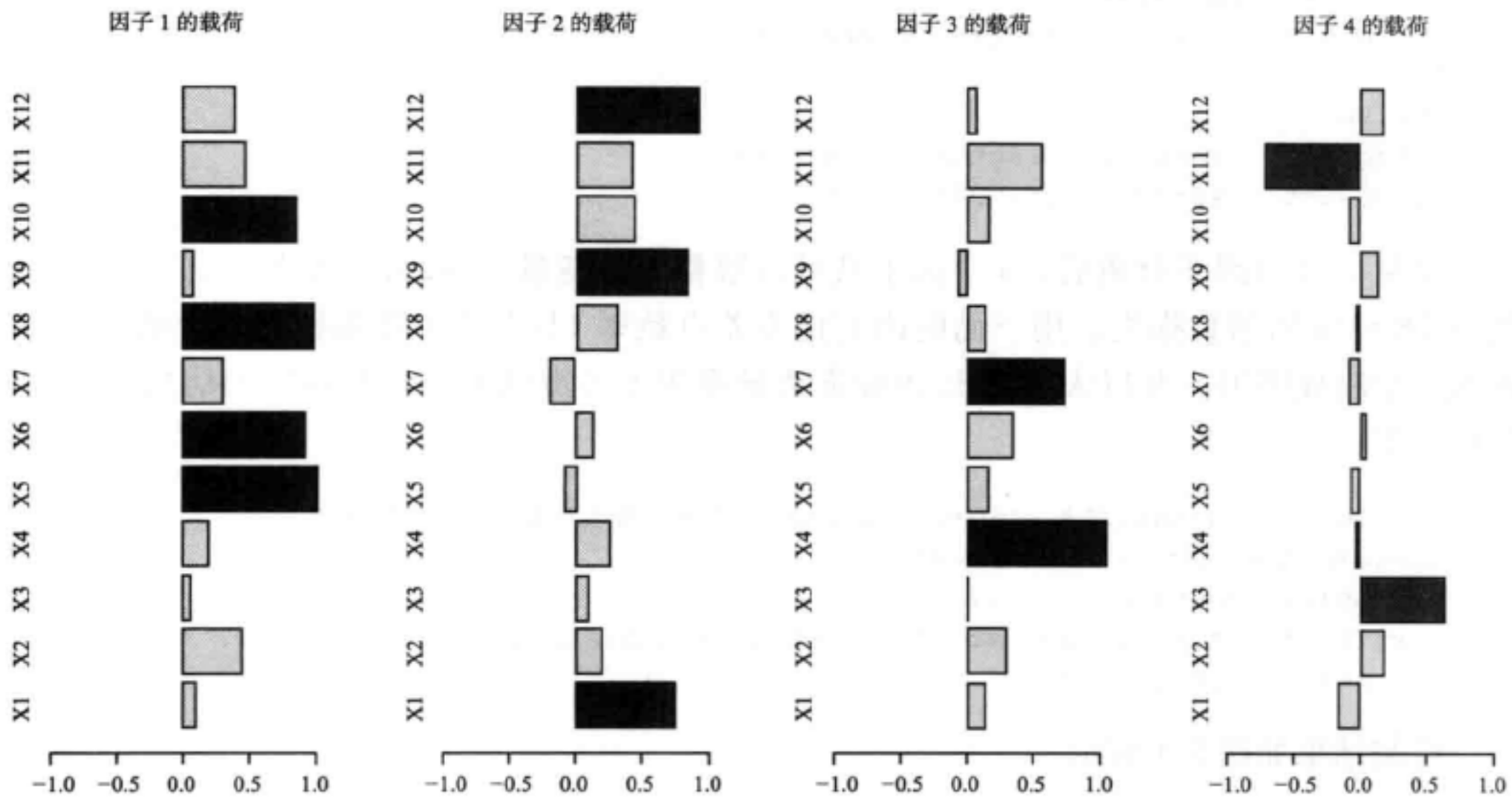


图 8-3 因子载荷结果

（1）方差贡献（特征值)(SS loadings)

根据方差贡献（特征值）的计算式：ssloading[i]= (a_{i1}^2+a_{i2}^2+..+$a_{in\text{-}1}^2$+a_{in}^2)，进行如下计算：

```
ssloadings<-sol$loadings[1,]
for(i in 1:factors){
    ssloadings[i]<-sum((sol$loadings[,i])^2)
}
ssloadings
  Factor1   Factor2   Factor3   Factor4
3.4892163 2.3335563 1.8690071 0.8787839
```

（2）方差贡献率（Proportion Var）

根据方差贡献率的含义（方差贡献率是各因子的方差贡献（特征值）与原始变量数目之比：pro.var[i]= ssloading[i]/n)，可以得到：

```
pro.var<-ssloadings/12
pro.var
   Factor1    Factor2    Factor3    Factor4
0.29076802 0.19446303 0.15575059 0.07323199
```

（3）累积方差贡献率（Cumulative Var）

根据累积方差贡献率的含义（累积方差贡献率是各因子方差贡献率的累积比例数值：cum.var[i]= cum.var[i-1]+pro.var[i])，可以得到：

```
cum.var= pro.var/sum(pro.var); factors=4
```

```
cum.var[1]= pro.var[1]
for(i in 2:factors){
    cum.var[i]<- cum.var[i-1]+ pro.var[i]
}
cum.var
  Factor1   Factor2   Factor3   Factor4
0.2907680 0.4852311 0.6409816 0.7142136
```

可见，经过因子分析后，4个因子共可以解释原始变量71.42%的方差，也就是说，大约有28.58%的信息损失。用下面的语句把方差贡献率（柱状图）和累积方差贡献率（蓝色曲线）绘制成图形，并以大于70%为标准衡量累积方差贡献率（小于30%的信息损失量）。代码如下：

```
barplot(ssloadings/12,ylim=c(0,1),main=" 各个因子对整体方差的贡献率 ")
lines(cum.var,col="blue",lwd=2)
points(c(1:factors), cum.var)
text(c(1:factors), cum.var +0.05,labels=round(cum.var*10000)/10000)
abline(h=0.7,col="red")
```

绘制结果如图8-4所示。

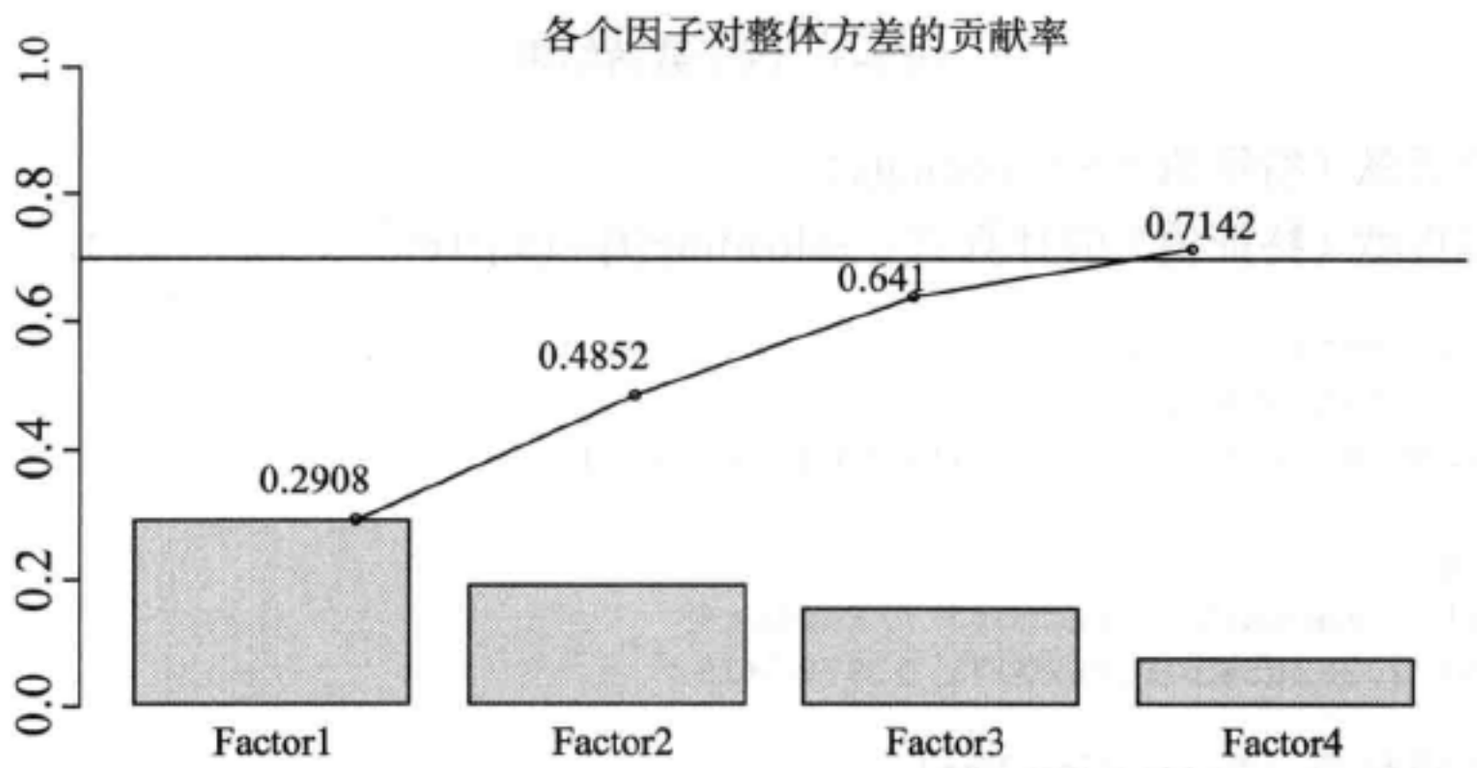


图8-4 方差累积贡献率结果

可见4个因子的累积方差贡献率大于70%，在可以容忍的信息损失量范围内。

7. 因子得分

由于前面在使用核心函数factanal时，把因子得分方法参数scores设置为了“Bartlett”，即使用加权最小二乘法做因子得分，因此可以直接使用sol$score来读取48个用户问卷调查结果数据经过因子分析后的降维数据。代码如下：

```
sol$score
        Factor1     Factor2     Factor3      Factor4
1   0.741109106  0.61737854 -0.76692351 -0.626287904
2   1.277279835  0.72782777  0.25177982  0.037167675
```

```
                    <中间省略>
47 -0.911328248 -1.98565627 -0.44024278  1.135117921
48 -0.756733576 -1.95797515 -1.24601905  0.606741687
```

8.2 聚类分析

聚类分析是典型的非监督分析，其分析的目的并不是对某一目标变量建模，而是对距离相近的样本进行归类（Q 型聚类）或对相近的变量指标进行归类（R 型聚类）。

8.2.1 距离

1. 连续型变量距离

（1）原理

对于连续型变量数据，有一些典型的距离定义，假设一组样本数据有 p 个变量指标和 n 组样本，具体见表 8-6。

表 8-6 数据示例

样本	变量 X_1	变量 X_2	…	变量 X_p
1	x_{11}	x_{12}	…	x_{1p}
2	x_{21}	x_{22}	…	x_{2p}
…	…	…	…	…
n	x_{n1}	x_{n2}	…	x_{np}

则常用的距离定义式如表 8-7 所示。

表 8-7 常用距离说明

距离	定义式	说明
绝对值距离	$d_{ij}(1)=\sum_{k=1}^{p}\lvert x_{ik}-x_{jk}\rvert$	绝对值距离是在一维空间下进行的距离计算
欧式距离	$d_{ij}(2)=\sqrt{\sum_{k=1}^{p}(x_{ik}-x_{jk})^2}$.	欧式距离是在二维空间下进行的距离计算
闵可夫斯基距离	$d_{ij}(q)=\left[\sum_{k=1}^{p}(x_{ik}-x_{jk})^q\right]^{1/q}$，$q>0$.	闵可夫斯基距离是在 q 维空间下进行的距离计算
切比雪夫距离	$d_{ij}(\infty)=\max_{1\le k\le p}\lvert x_{ik}-x_{jk}\rvert$.	切比雪夫距离是 q 取正无穷大时的闵可夫斯基距离，即切比雪夫距离是在 $+\infty$ 维空间下进行的距离计算
Lance 距离	$d_{ij}(L)=\sum_{k=1}^{p}\frac{\lvert x_{ik}-x_{jk}\rvert}{x_{ik}+x_{jk}}$	减弱极端值的影响能力
归一化距离	$d_{ij}=\sum_{k=1}^{p}\frac{\lvert x_{ik}-x_{jk}\rvert}{\max(x_k)-\min(x_k)}$	自动消除不同变量间的纲量影响，其中每个变量 k 的距离取值均是 [0,1]

（2）R 语言实现

在 R 语言中，使用 dist 函数可以把一个矩阵或数据框转化为距离矩阵，dist 函数的参数如表 8-8 所示。

表 8-8 dist 函数的参数

<table>
<tr><th>参数</th><th>说明</th></tr>
<tr><td>x</td><td>矩阵或数据框形式的样本数据，其中每行为一个样本数据，每列为一个指标变量。例如，data<-matrix(1:20,c(4,5)) 是一个具有 5 个指标、4 个样本的数据，其格式如下：
[,1] [,2] [,3] [,4] [,5]
[1,] 1 5 9 13 17
[2,] 2 6 10 14 18
[3,] 3 7 11 15 19
[4,] 4 8 12 16 20</td></tr>
<tr><td>method</td><td>距离的计算方法：
manhattan 表示绝对值距离；euclidean 表示欧式距离（默认取值）；
minkowski 表示闵可夫斯基距离；maximum 表示切比雪夫距离；
canberra 表示 Lance 距离；binary 表示 binary 距离，具体定义参考上面对分类数据距离的说明</td></tr>
<tr><td rowspan="3">diag</td><td colspan="2">为 FALSE（默认取值）时，不计算对角线上（即某个样本与自己的距离）的距离。
为 TRUE 时，计算对角线上的距离</td></tr>
<tr><td>dister<-dist(data);dister
1 2 3
2 2.236068
3 4.472136 2.236068
4 6.708204 4.472136 2.236068</td><td>dister<-dist(data,diag=T);dister
1 2 3 4
1 0.000000
2 2.236068 0.000000
3 4.472136 2.236068 0.000000
4 6.708204 4.472136 2.236068 0.000000</td></tr>
<tr><td rowspan="3">upper</td><td colspan="2">为 FALSE（默认取值）时，只以下三角形式输出
为 TRUE 时，以上三角形式和下三角形式输出</td></tr>
<tr><td>dister<-dist(data,upper=T);dister
1 2 3 4
1 2.236068 4.472136 6.708204
2 2.236068 2.236068 4.472136
3 4.472136 2.236068 2.236068
4 6.708204 4.472136 2.236068</td><td>dister<-dist(data,diag=T,upper=T);dister
1 2 3 4
1 0.000000 2.236068 4.472136 6.708204
2 2.236068 0.000000 2.236068 4.472136
3 4.472136 2.236068 0.000000 2.236068
4 6.708204 4.472136 2.236068 0.000000</td></tr>
</table>

（3）连续型变量距离的去量纲处理

对连续型变量的距离计算，实际存在一个量纲问题。所谓量纲，就是指标的单位。例如，同一个人的身高既可以写为 1.75m，也可以写为 175cm，其中身高指标米 / 厘米就是量纲。先来看个例子，表 8-9 中是两个用户的日网站停留时间变量（变量 *X*1.1 的单位是秒，变量 *X*1.2 的单位是分）和用户日访问页面数变量（变量 *X*2 的单位是页）数据。

表 8-9 数据示例

样本	*X*1.1 用户日网站停留时间（秒）	*X*1.2 用户日网站停留时间（分）	*X*2 用户日访问页面数
1	120	2	3
2	180	3	5

同样是使用欧式距离计算样本 1 和样本 2 的距离，当“用户日网站停留时间”变量分别以“秒”和“分”为单位时，其距离的结果会相差很多，绘制代码如下：

```
data.sec<-matrix(c(120,180,3,5),c(2,2))
data.min<-matrix(c(2,3,3,5),c(2,2))
par(mfcol=c(1,2))
plot(data.sec,pch=16,main=paste("用户日网站停留时间单位是秒时的距离：",dist(data.sec)))
plot(data.min,pch=16,main=paste("用户日网站停留时间单位是分时的距离：",dist(data.min)))
```

绘制结果如图 8-5 所示。以“秒”为单位时的距离为 60.03332，以“分”为单位时的距离为 2.236068。

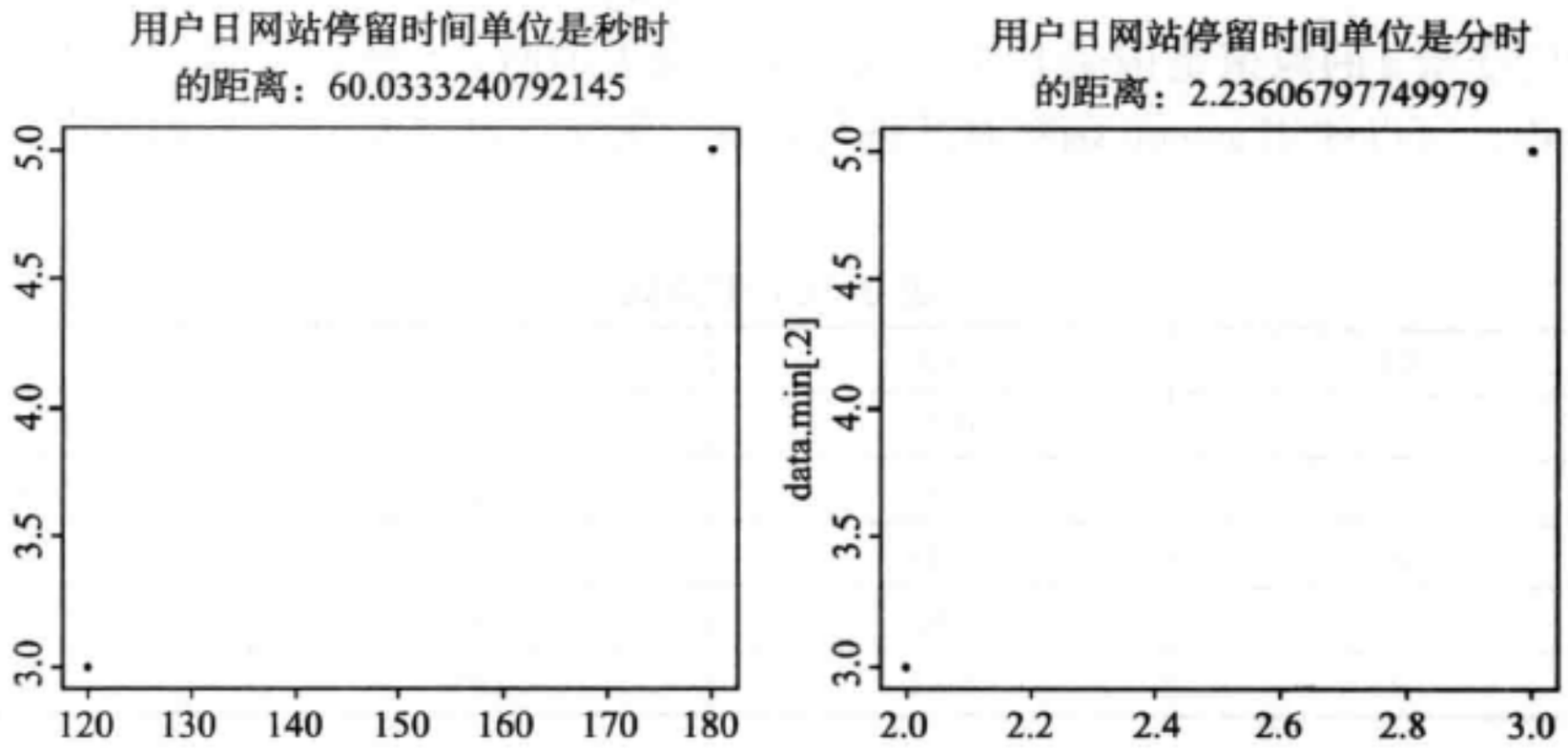


图 8-5 未去量纲结果

此时需要对数据使用 scale 函数进行标准化处理，以消除变量间的纲量效应影响，公式如下：

$z_k=x_k-\text{mean}(x)/\text{sd}(x)$

其中 mean(x) 是变量 x 的样本均值，sd(x) 是样本标准差。转换后的变量 z 是均值为 0，标准差为 1 的数据。进行标准化处理的绘制语句如下：

```
data.sec<-matrix(c(120,180,3,5),c(2,2)); data.sec<-scale(data.sec)
data.min<-matrix(c(2,3,3,5),c(2,2)) ; data.min <-scale(data.min)
par(mfcol=c(1,2))
plot(data.sec,pch=16,main=paste("用户日网站停留时间单位是秒时的距离：",dist(data.sec)))
plot(data.min,pch=16,main=paste("用户日网站停留时间单位是分时的距离：",dist(data.min)))
```

绘制结果如图 8-6 所示。经过标准化处理后，不论单位是分还是秒，其距离均为 2。

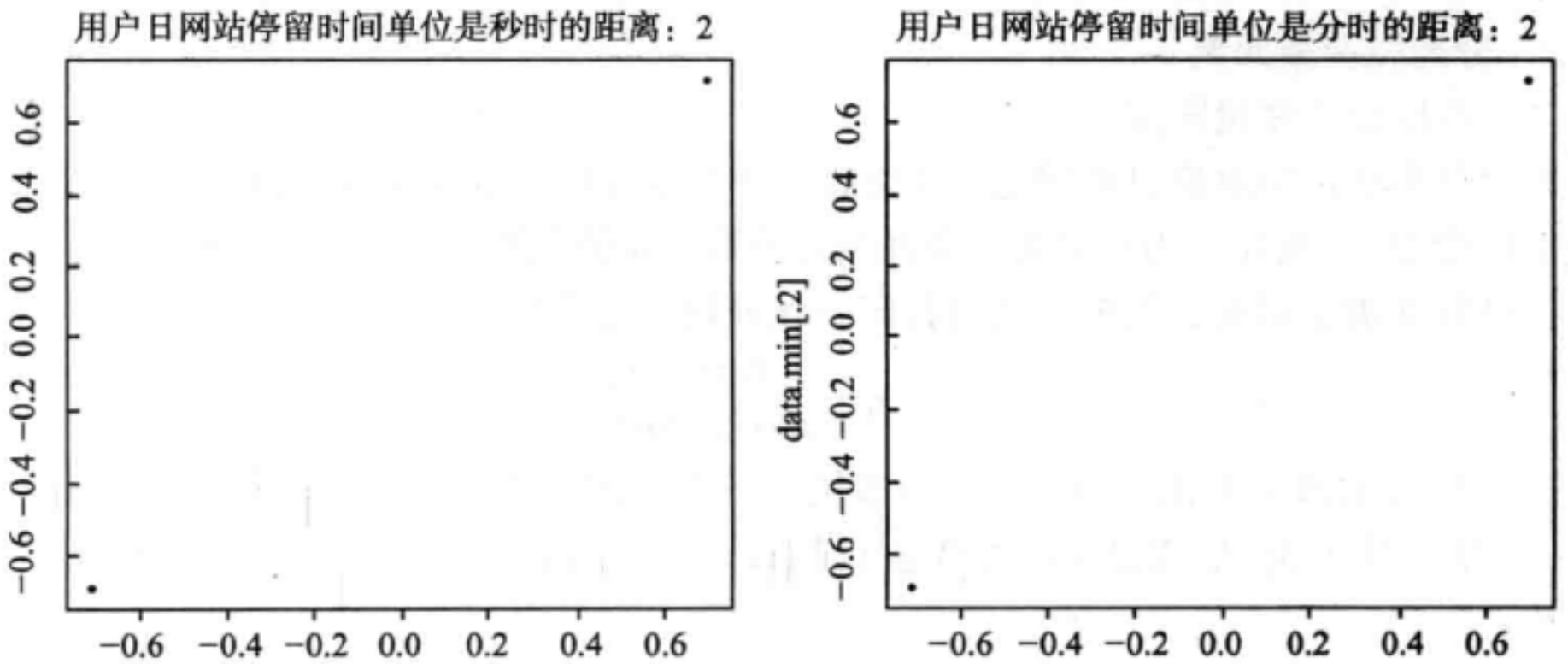


图 8-6 去量纲结果

当然，也可以不使用标准化处理，而是使用归一化处理，公式如下：

$$z_k=\frac{X_k\text{-}\min(x)}{\max(x)\text{-}\min(x)}$$

转化后变量 z 的取值范围是 [0,1]，其中 x 取最大值时，z 为 1，x 取最小值时，z 为 0。在 R 语言中，可以使用 sweep 函数对其进行计算。例如，要对表 8-10 中的数据进行归一化处理。

表 8-10 数据集

	X1	X2	X3	X4
1	1	0.21	−109	80001
2	3	0.11	120	90881
3	4	0.13	101	89917
4	5	0.901	298	55672
5	0	0.41	293	18792

代码如下：

```
    x<-data.frame(x1=c(1,3,4,5,0),x2=c(0.21,0.11,0.13,0.901,0.41),x3=
c(-109,120,101,298,293)
                      ,x4=c(80001,90881,89917,55672,18792))
    center <- sweep(x, 2, apply(x, 2, min))
    R <- apply(x, 2, max) - apply(x,2,min)
    x_star <- sweep(center, 2, R, "/"); x_star
       x1         x2        x3        x4
    1 0.2  0.12642225  0.0000000  0.8490754
    2 0.6  0.00000000  0.5626536  1.0000000
    3 0.8  0.02528445  0.5159705  0.9866276
    4 1.0  1.00000000  1.0000000  0.5115898
    5 0.0  0.37926675  0.9877150  0.0000000
```

由结果可知，变量 $x1 \sim x4$ 的最大值恒为 1，最小值恒为 0，即不论原始变量的单位是什么，变化后一定处于 [0,1] 内。

2. 分类型变量距离

（1）有序分类变量距离

有序分类变量的取值只有固定值（类似于分类型变量），但是其数值又有大小含义（类似于连续型变量）。例如，用户对某一商品的评分数（取值范围是 1 ～ 5）变量即为有序分类变量。在计算此类变量前，先要对其进行归一化处理，这类似于连续型变量的去纲量化处理。

$$z_k=\frac{X_k\text{-}\min(x)}{\max(x)\text{-}\min(x)}$$

转化后 z_k 的取值范围是 [0，1]，一般情况下把有序分类变量 x 的可选取值规定为 1，2，3，…，M_x。其中 M_x 是变量 x 的可取值状态数，即 x 的最大取值。这样上面的公式就可以简化为：

$$z_k=\frac{X_k\text{-}1}{M_x\text{-}1}$$

在完成数据归一化处理后，即可视有序分类变量为连续型变量，并计算其距离。

（2）二元分类变量距离

一个二元分类变量的取值仅为 0 或 1。例如，表 8-11 的 4 个用户变量即为二元分类变量。

表 8-11　二元分类变量数据示例

样本	性别（男 / 女）	国内 / 外用户（国内用户 / 国外用户）	VIP/ 非 VIP 用户（VIP 用户 / 非 VIP 用户）	完成第 1 次购买（完成 / 未完成）
1	男	国内用户	VIP	完成
2	女	国内用户	非 VIP	未完成

其中，“性别”和“国内 / 外用户”两个变量取值的重要程度相同，因此说它们是对称二元分类变量。而“ VIP/ 非 VIP 用户”变量中取值为 VIP 的重要性要比非 VIP 大很多，因此称其为非对称二元分类变量。同样“完成第 1 次购买”变量中取值为完成的重要性也要比未完成的重要性大很多，所以它也是非对称二元分类变量。

由二值数据计算差异矩阵来看，如果认为所有二值变量的权值均相同，就能得到如表 8-12 所示的 2×2 条件表。

表 8-12　条件表

	样本 i 数据		
样本 j 数据	取值	1	0
	1	m1	m10
	0	m01	m0

其中 $m1$ 是取值均为 1（1 配对）的变量数，$m0$ 是均为 0（0 配对）的变量数，$m10+ m01=m2$ 是取值不同（不配对）的变量数，则对称二元分类变量的距离可以计算为：

$$d_{ij}=\frac{m_2}{m_1+m_0+m_2}$$

d_{ij} 的取值范围是 [0,1]，如果所有变量均可以匹配（1 配对或者是 0 配对），则 $m2=0$，此时距离最小，等于 0。如果所有变量的均不匹配，则 $m1=m0=0$，此时距离最大，为 $m2/m2=1$。

例如，针对表 8-12 中的“性别”（男为 1，女为 0）和“国内 / 外用户”（国内用户为 1，国外用户为 0）两个变量，样本 1 和样本 2 的 2×2 条件表如表 8-13 所示。

表 8-13　条件表

	样本 1 数据		
样本 2 数据	取值	1	0
	1	m1=1	m10=0
	0	m01=1	m0=0

则距离 $d_{12}=（1+0）/（1+0+1+0）=0.5$

非对称二元分类变量的距离可以通过如下公式计算：

$$d_{ij}=\frac{m_2}{m_1+m_2}$$

注意，对于非对称变量，在分母中不再考虑匹配重要较低因素的数目 $m0$。d_{ij} 的取值范围是 [0,1]，如果所有变量均可以匹配（1 配对或者是 0 配对），则 $m2=0$，此时距离最小，等于 0。如果所有变量均不匹配 1（重要性较高的取值），则 $m1=0$，此时距离最大，为 $m2/$

$m2$=1。例如，针对表 8-12 中的“VIP/ 非 VIP 用户”（VIP 为 1，非 VIP 为 0）和“完成第 1 次购买”(完成为 1，未完成为 0）两个变量，样本 1 和样本 2 的 2×2 条件表如表 8-14 所示)。

则距离 d_{12}=（1+0）/（1+0+0）=1

（3）无序分类变量距离

无序分类变量可以把每个 p 元变量转化为 p 个非对称二元变量，并使用相同的样本数来计算距离。无序分类变量数据示例如表 8-15 所示。

表 8-14 条件表

	样本 1 数据		
样本 2 数据	取值	1	0
	1	m1=0	m10=0
	0	m01=1	m0=1

表 8-15 无序分类变量数据示例

样本	职业（教师 / 工人 / 工程师）	网龄（小于 1 年 /1-2 年 / 大于 2 年）
1	教师	1-2 年
2	教师	小于 1 年

把表 8-15 中的数据转化为非对称二元分类变量，如表 8-16 所示。

表 8-16 非对称二元分类变量数据示例

样本	职业			网龄		
	教师	工人	工程师	小于 1 年	1-2 年	大于 2 年
1	1	0	0	0	1	0
2	1	0	0	1	0	0

然后统计转化后各二元变量均为 1（1 配对）的数目 $m1$，以及取值不同（不配对）的数目 $m2$。例如，在表 8-16 中，职业是否为“教师”这一变量的取值为 1 时，$m1$=1，$m2$=2。

最后给出样本 i 和样本 j 的距离定义，如下：

$$d_{ij}=\frac{m_2}{m_1+m_2}$$

可见，样本 1 和样本 2 的距离 $d_{12}=m2/(m1+m2)=2/(1+2)=2/3$。在 R 语言中，可以使用 dist 函数并把 method 参数设置为“binary”来计算无序分类变量的距离。

注意

以上距离只适用于所有变量均是连续型或均是分类型的情况。对于混合型的样本数据，可以考虑分别计算连续型和分类型的距离，最后再汇成综合距离。

3. 混合变量处理

上面介绍的连续型变量、有序分类变量、对称二元分类变量、非对称二元分类变量，以及无序分类变量的距离计算只适用于所有变量均是以上类型的情况。但在实际数据库中，数据对象往往是用复合数据类型来描述的，而且它们（同时）常常包含上述几种数据类型。一个更好的处理方法就是将所有类型的变量距离全部映射到 [0,1] 范围内，然后把它们放在一起进行综合叠加计算。假设一个数据集包含 p 个组合类型变量，则对象 i 和对象 j 之间的距

离 d_{ij} 就可以定义为：

$$d(i,j)=\frac{\sum_{f=1}^{p}\delta_{ij}^{(f)}d_{ij}^{(f)}}{\sum_{f=1}^{p}\delta_{ij}^{(f)}}$$

如果 x_{if} 是指第 f 个变量中的第 i 个样本数据值，则：

- 如果 x_{if} 和 x_{jf} 均不存在，则 $\delta_{ij}^{(f)}=0$。
- 如果第 f 个变量是非对称二元分类变量，且 $x_{if}=x_{jf}=0$，则 $\delta_{ij}^{(f)}=0$。
- 其余情况下，$\delta_{ij}^{(f)}=1$。

$d_{ij}^{(f)}$ 可以使用如表 8-17 所示的方式计算。

表 8-17 计算说明

变量类型	计算 $d_{ij}^{(f)}$
连续型变量	$d_{ij}^{(f)}=\frac{\lvert x_{if}-x_{jf}\rvert}{\max(x_{if})-\min(x_{if})}$
有序分类变量	先使用 $z_k=\frac{x_k-1}{M_x-1}$ 把第 f 个变量数据转化为连续型变量，再使用 $d_{ij}^{(f)}=\frac{\lvert x_{if}-x_{jf}\rvert}{\max(x_{if})-\min(x_{if})}$ 进行计算
对称二元分类变量、 无序分类变量，以及非对称二元分类变量	如果 $x_{if}=x_{jf}$，则 $d_{ij}^{(f)}=\frac{m_2}{m_1+m_0+m_2}=0/(m1+m0+0)=0$ 如果 $x_{if}!=x_{jf}$，则 $d_{ij}^{(f)}=\frac{m_2}{m_1+m_0+m_2}=m2/(0+0+m2)=1$

下面来看一个实例，实例涉及的数据见表 8-18。

表 8-18 数据集

	网站停留时间	访问页面数	用户等级	性别	国内 / 外用户	VIP/ 非 VIP	完成第 1 次购买	职业	网龄
1	120	3	1	1(男)	1(国内用户)	1(VIP)	0(未完成)	1(教师)	2(1 ～ 2 年)
2	180	5	2	0(女)	1(国内用户)	0(非 VIP)	0(未完成)	1(教师)	1(小于 1 年)
3	30	1	1	0(女)	0(国外用户)	1(VIP)	1(完成)	2(工人)	2(1 ～ 2 年)
4	900	21	5	0(女)	0(国外用户)	1(VIP)	0(未完成)	3(工程师)	3(大于 2 年)

数据准备：

```
x1<-c(120,180,30,900);x2<-c(3,5,1,21);
x3<- c(1,2,1,5);
x4<- c(1,0,0,0); x5<- c(1,1,0,0);
x6<- c(1,0,1,1); x7<- c(0,0,1,0);
x8<- c(1,1,2,3); x9<- c(2,1,2,3)
```

$x1$ 和 $x2$ 是连续型数据，使用 c() 来建立一个连续型序列；$x3$ 是多元有序分类变量，本应该使用 as.ordered 建立有序因子序列，但由于后续在计算距离时要转化为连续型数据，所以这里直接使用 c() 来建立一个连续型序列；$x4$ 至 $x7$ 是二元分类变量 ($x4$ 和 $x5$ 是对称二元

变量，$x6$ 和 $x7$ 是非对称二元变量），直接使用 as.factor 建立无序因子序列；$x8$ 和 $x9$ 是多元无序分类变量，使用 as.factor 建立。最后将这些序列放于数据框 data 中。

计算连续型变量 $x1$ 和 $x2$ 的距离，代码如下：

```
dis.numeric<-function(x.vector){
    x.rank<-max(x.vector)- min(x.vector)
    dx<- as.matrix(dist(x.vector,method="manhattan", diag=T)/ x.rank);
    dx.step<- matrix(1,ncol=length(x.vector), nrow=length(x.vector))
    list(dx=dx, dx.step= dx.step)
}
di1<-dis.numeric(x1)
di2<-dis.numeric(x2)
```

dis.numeric() 函数是自定义的计算连续型变量距离的函数，输入的变量 x.vector 必须是一个连续型向量。输出变量是一个 list，其 dx 代表 d_{if}，dx.step 代表$\delta_{ij}^{(f)}$。计算 d_{if} 的核心公式如下：

$$d_{ij}^{(f)}=\frac{|x_{if}-x_{jf}|}{\max(x_{if})-\min(x_{if})}$$

首先使用 dist(x.vector，method="manhattan") 计算出各个样本的绝对值差，再除以变量极差 max(x.vector)- min(x.vector)。对于连续型变量，因为其$\delta_{ij}^{(f)}$恒为 1，所以 dx.step 是与 dx 有相同行列数目，但是值均为 1 的矩阵。代码实现如下：

```
dx.step<- matrix(1,ncol=length(x.vector),nrow=length(x.vector))
```

计算有序分类变量 $x3$ 的距离，代码如下：

```
x3.z<- (x3-1)/(5-1)
di3<-dis.numeric(x3.z)
```

x3.z<- (x3-1)/(5-1) 语句的依据为：

$$z_k=\frac{x_k-1}{M_x-1}$$

将有序分类变量 $x3$ 转化为归一化的连续型数据 $x3.z$。这里的用户等级分为 5 个级别，即 M_x=5。然后计算连续型变量 $x3.z$ 的距离 d_{i3}。

计算二元变量 $x4$ 至 $x7$ 以及无序分类变量 $x8$ 和 $x9$ 的距离，代码如下：

```
dis.factor<-function(x.vector,blance=TRUE){
    x.num<-length(x.vector)
    x.matrix<-matrix(rep(x.vector,x.num),ncol=x.num ,byrow=T)
    x.matrix.t<-t(x.matrix)
    equal.matrix<- as.numeric(x.matrix!= x.matrix.t);
    dx<-matrix(equal.matrix,ncol= x.num);
    if(blance){
        dx.step<- matrix(1,ncol=length(x.vector),nrow=length(x.vector))
    }else{
        equal0.matrix<-as.numeric((x.matrix+x.matrix.t)!=0);
```

```
        dx.step<- matrix(equal0.matrix,ncol= x.num);
    }
    list(dx=dx, dx.step= dx.step)
}
di4<- dis.factor(x4);
di5<- dis.factor(x5);
di6<- dis.factor(x6,blance=F);
di7<- dis.factor(x7,blance=F);
di8<- dis.factor(x8);
di9<- dis.factor(x9);
```

dis.factor() 函数是自定义的用于计算无序分类变量以及二元变量距离的函数。如果参数 blance 设置为 FALSE，则计算非对称二元变量的距离，否则计算无序分类变量以及对称二元变量的距离。输出变量是一个 list，其 dx 代表 d_{if}，dx.step 代表$\delta_{ij}^{(f)}$。

使用下面的关系式来计算 d_{if}:

如果 x_{if} 等于 x_{jf}，则等于 0；如果 x_{if} 不等于 x_{jf}，则等于 1。

代码如下：

```
x.num<-length(x.vector)
x.matrix<-matrix(rep(x.vector,x.num),ncol=x.num ,byrow=T)
x.matrix.t<-t(x.matrix)
equal.matrix<- as.numeric(x.matrix!= x.matrix.t);
dx<-matrix(equal.matrix,ncol= x.num);
```

因为无序分类变量以及对称二元变量的$\delta_{ij}^{(f)}$恒为 1，所以 dx.step 是与 dx 有相同行列数，但值均为 1 的矩阵。代码如下：

```
dx.step<- matrix(1,ncol=length(x.vector),nrow=length(x.vector))
```

对于非对称二元变量，如果 x_{if} 等于 x_{jf} 且等于 0，则$\delta_{ij}^{(f)}$=0。代码如下：

```
equal0.matrix<-as.numeric((x.matrix+x.matrix.t)!=0);
dx.step<- matrix(equal0.matrix,ncol= x.num);
```

计算混合距离时，可使用：

$$d(i,j)=\frac{\sum_{f=1}^{p}\delta_{ij}^{(f)}d_{ij}^{(f)}}{\sum_{f=1}^{p}\delta_{ij}^{(f)}}$$

来计算最后的综合距离。代码如下：

```
d<-di1$dx+ di2$dx+ di3$dx+ di4$dx+ di5$dx+ di6$dx+ di7$dx+ di8$dx+ di9$dx
d<-d/(di1$dx.step+di2$dx.step+di3$dx.step+di4$dx.step+di5$dx.step+di6$dx.step+
di7$dx.step+di8$dx.step+di9$dx.step)
```

注意：因为$\delta_{ij}^{(f)}$等于 0 时，$\delta_{ij}^{(f)}$也等于 0（x_{if} 等于 x_{jf} 且等于 0 时），所以这里直接使用$\sum_{f=1}^{p}d_{ij}^{(f)}$代替分子$\sum_{f=1}^{p}\delta_{ij}^{(f)}d_{ij}^{(f)}$，最后得到综合后的距离 d。

```
d
          1         2         3         4
1 0.0000000 0.4273707 0.4670498 0.8495690
2 0.4273707 0.0000000 0.6247126 0.7971983
3 0.4670498 0.6247126 0.0000000 0.6666667
4 0.8495690 0.7971983 0.6666667 0.0000000
```

4. 相似度距离

前面介绍的距离实际是以样本数据的差异为基础来衡量的，相似度距离则是以样本数据的相似程度来衡量的。相似度距离还有一个典型的应用：通过计算变量间的相似度距离来对变量进行聚类，即完成 R 型聚类。

常用度量相似度距离的方法有夹角余弦和相关数两种。

（1）夹角余弦

如果两组数据（可以是两个样本数据或两个变量数据）*Xi* 和 *Xj* 的数据分别为 $(x_{1i},x_{2i},..,x_{ni})$ 和 $(x_{1j},x_{2j},..,x_{nj})$，则两组数据的夹角余弦相似度 $\cos_{ij}$ 定义为：

$$\cos_{ij}=\frac{\sum_{k=1}^{n}x_{ki}x_{kj}}{\sqrt{\sum_{k=1}^{n}x_{ki}^2}\sqrt{\sum_{k=1}^{n}x_{kj}^2}} \quad i,j=1,2,\cdots,p.$$

$\cos_{ij}$ 处于 [−1,1] 范围内，当 $\cos_{ij}$ 趋近于 ±1 时，表明两组数据有较强的相关性。当 $\cos_{ij}$ 趋近于 0 时，表明两组数据无任何相关性。夹角余弦相似度的自定义函数如下：

```
cosij<-function(xi,xj){
    sum(xi*xj)/(sqrt(sum(xi^2))*sqrt(sum(xj^2)))
}
```

如果 *xi* 和 *xj* 的长度为 2，则可以视其为二维平面上的两个点。例如，分别设置 *xi*=*c*(1,2), *xj*=*c*(3,4); *xi*=*c*(3,4), *xj*=*c*(−4,3); *xi*=*c*(1,2), *xj*=*c*(−2,−4)。然后计算其夹角余弦相似度，并绘制原点与 *xi* 和 *xj* 组成的向量。代码如下：

```
plot.cos<-function(xi,xj){
    plot(c(xi[1],xj[1]), c(xi[2],xj[2]),pch=16,main=paste("夹角余弦相似度=",
    cosij(xi,xj)),xlim=c(-5,5),ylim=c(-5,5))
    abline(v=0);abline(h=0)
    arrows(0,0,xi[1],xi[2])
    arrows(0,0,xj[1],xj[2])
}
par(mfcol=c(1,3))
xi=c(1,2);xj=c(3,4); plot.cos(xi,xj)
xi=c(3,4);xj=c(-4,3); plot.cos(xi,xj)
xi=c(1,2);xj=c(-2,-4); plot.cos(xi,xj)
```

绘制结果如图 8-7 所示。

观察左侧图形，夹角余弦相似度越趋于 1（此时等于 0.984），夹角越小，可以解释为其正相关性越大。夹角余弦相似度等于 0 时（中间图形），向量是垂直的，可以解释为完全不

相关。等于 −1 时（右侧图形），向量是反向平行的，可以解释为完全负相关。同理，如果 *xi* 和 *xj* 的数据长度为 p，则可以推广至 p 维空间。

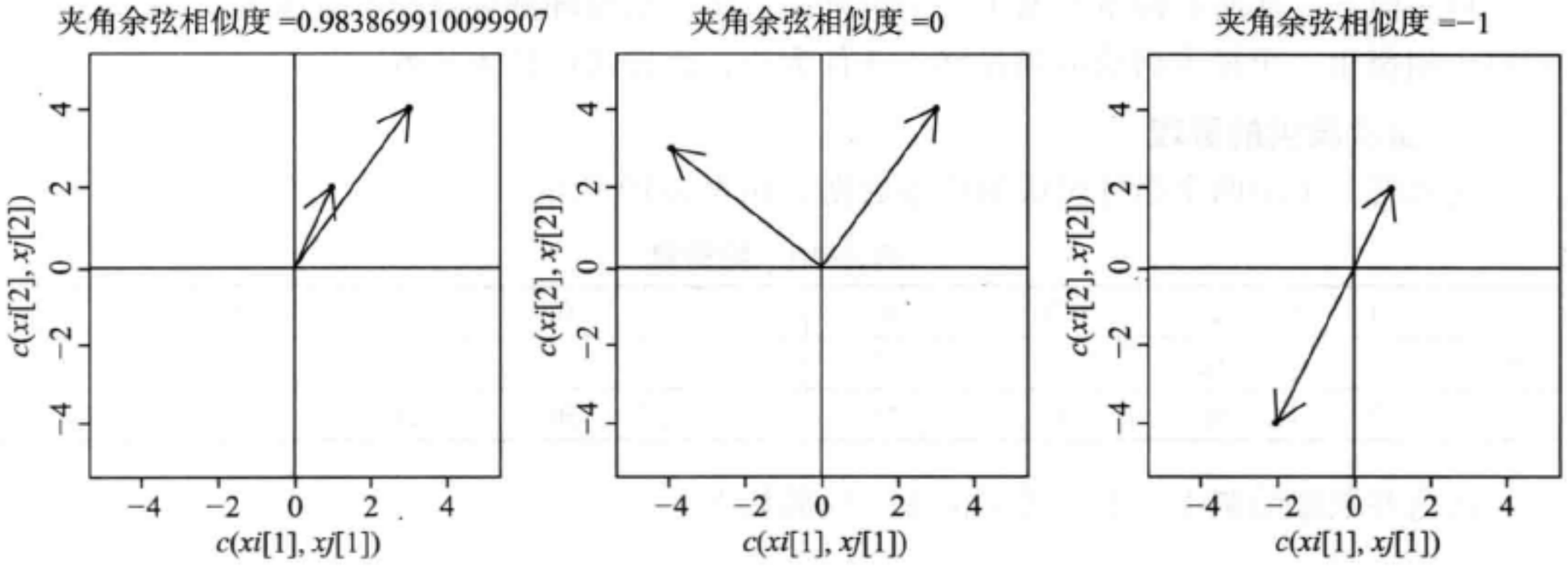


图 8-7 余弦相似度示意图

（2）相关系数

在 5.5.1 节中曾介绍过数据 *xi* 和数据 *xj* 的相关性系数 r_{ij} 的定义：

$$r_{xixj}=\frac{\sum_{k=1}^{n}(x_{ki}-\overline{x}_i)(x_{kj}-\overline{x}_j)}{\sqrt{\sum_{k=1}^{n}(x_{ki}-\overline{x}_i)^2}\sqrt{\sum_{k=1}^{n}(x_{kj}-\overline{x}_j)^2}} \quad i,j=1,2\cdots,p.$$

对于该系数，可直接使用 cor() 函数来计算。

（3）相似度与距离的转换

在上面相似系数的定义中可以看出，取值范围均是 [−1,1]，但这类值并不可以直接当成距离来使用。

如果要求相似度的绝对值越大（趋近于 1），距离越小，绝对值越小（趋近于 0），距离越大，则可以定义距离 d_{ij}=sqrt(1−c_{ij}^2) 或 d_{ij}= sqrt(1−r_{ij}^2)。其中 d_{ij} 是距离，c_{ij} 和 r_{ij} 分别是两种相似度，此时 d_{ij} 的取值范围是 [0,1]。

如果要求相似度越大（趋近于 1），距离越小，相似度越小（趋近于 −1），距离越大，则可以定义距离 d_{ij}=−log ((*c*$_{ij}$/2)+0.5) 或 d_{ij}=−log ((*r*$_{ij}$/2)+0.5) 。其中 d_{ij} 是距离，c_{ij} 和 r_{ij} 分别是两种相似度，此时 d_{ij} 的取值范围是 [0,+ ∞]。

注意

使用相似度描述的距离常常用于计算两个变量间的距离，并完成 R 型聚类。

8.2.2 层次聚类

层次聚类通过分解给定的数据集来创建一个层次。根据层次分解形成的方式，可以将层次方法分为自下而上和自上而下两种类型，有些类似于决策树的生长和停止生长。

自下而上：从每个样本均为一个（单独的）组开始，逐步将这些样本组合并，直到合并到层次顶端或满足终止条件为止，此方法也称为聚合。

自上而下：从所有样本均属于一个组开始，每一次循环都会将样本分解为更小的组，直到每个组都由一个样本构成或满足终止条件为止，此方法也称为分解。

1. 层次聚类的原理

先来看一个由两个指标组成的样本数据，如表 8-19 所示。

表 8-19 数据集

	1	2	3	4	5	6	7	8	9
*X*1	1	12	3.3	5	2	14	5.5	1.1	10
*X*2	2	45	1.2	21	2.3	50	20	3.4	40

把这些数据绘制于一个二维平面中。代码如下：

```
data<-data.frame(x1=c(1,12,3.3,5,2,14,5.5,1.1,10),x2=c(2,45,1.2,21,2.3,50,20,
3.4,40))
plot(data,pch=16)
```

绘制结果如图 8-8 所示。

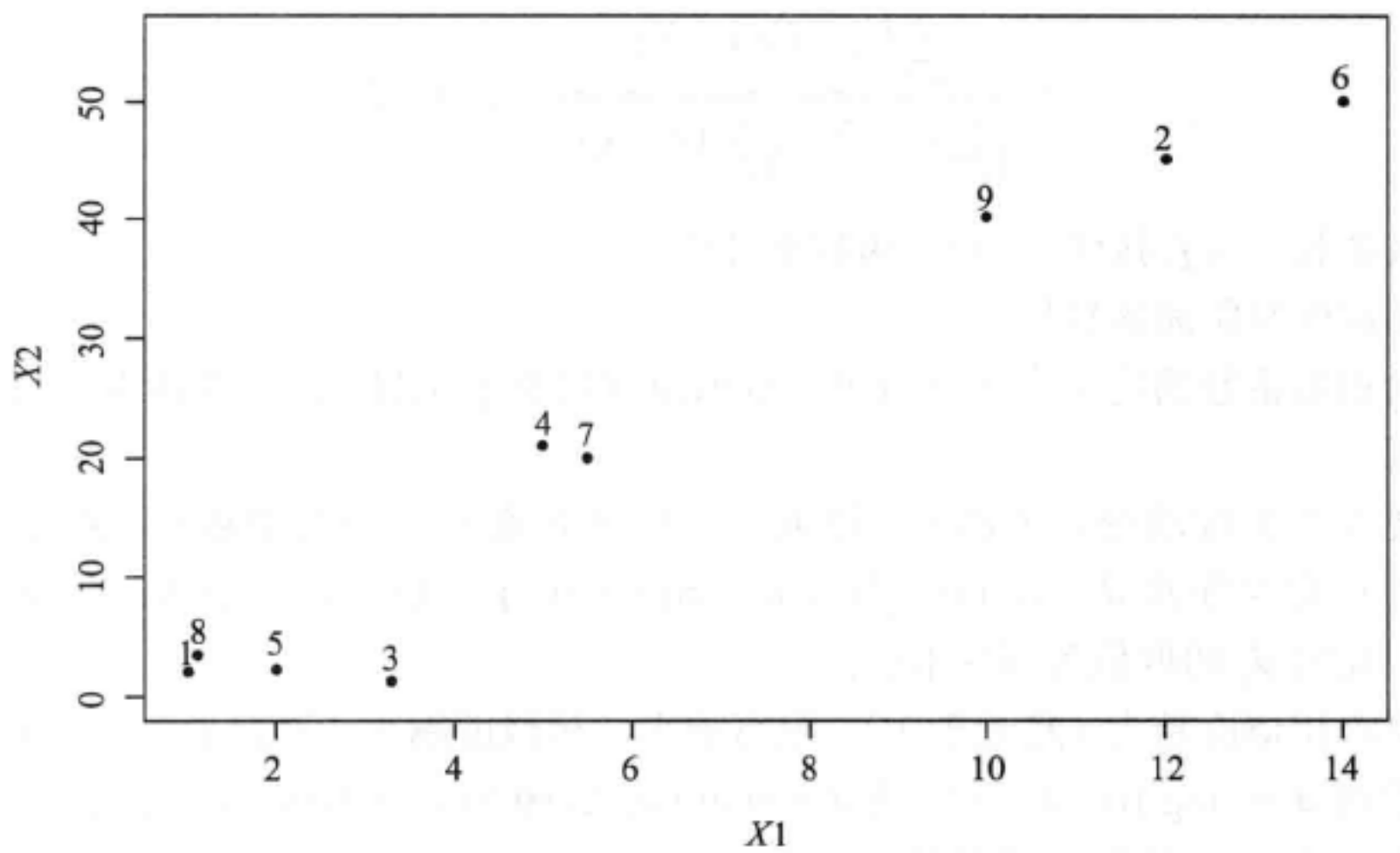


图 8-8 散点图

可以看到，图 8-8 中的 9 个点分成了 3 组，分别是由 1、8、5、3 组成的第 1 组；由 4、7 组成的第 2 组；由 9、2、6 组成的第 3 组。当然如果想继续细分，也可以把第三组的点进一步分离，这取决于分组的精细颗粒度。以上不过是通过人为观察得到的结果，一旦数据量增加以及变量维度增加，就无法通过人为观察得到正确的结果了。这时需要使用一些技术和技巧来帮助划分比如，使用层次方法来实现上面的聚类。

自下而上（最小距离聚合法）：这里以最小距离来衡量两组（点）是否要进行聚合，对于

两个聚类组 C1 和 C2，若 C1 中的样本与 C2 中样本间的欧式距离为不同聚类中任两个样本间的最小距离，则 C1 和 C2 就可以进行聚合。可重复使用上述方法，直到所有聚类最终聚合为一个聚类为止。

在 R 语言中，使用 hclust 函数进行层次聚类，其基本形式是 hclust(d, method = "complete")，参数说明如表 8-20 所示。

表 8-20 hclust 函数参数说明

参数	说明
d	有 dist 结构的数据对象，是可以直接使用 dist() 函数输出的对象，也可以通过 as.dist() 函数把数据框或矩阵转化为 dist 结构
method	层次聚类的方法有以下几种： single：最小距离法；complete：最大距离法；median：中间距离法 mcquitty：Mcquitty 相似法；average：类平均距离法；centroid：重心法； ward：离差平方和法 以上各种方法的公式请参考《统计建模与 R 软件》一书的 8.2.2 节。

下面对本节的示例使用最小距离的聚合法。代码如下：

```
d<-dist(data)
h.single.sol<-hclust(d,method="single")
par(mfrow = c(1, 3))
plclust(h.single.sol, hang=-1,main="最小距离的聚合聚类")
plclust(h.single.sol, hang=-1,main="最小距离的聚合聚类(3组)")
re3<-rect.hclust(h.single.sol, k=3)
plclust(h.single.sol, hang=-1,main="最小距离的聚合聚类(5组)")
re5<-rect.hclust(h.single.sol, k=5)
```

绘制结果如图 8-9 所示。

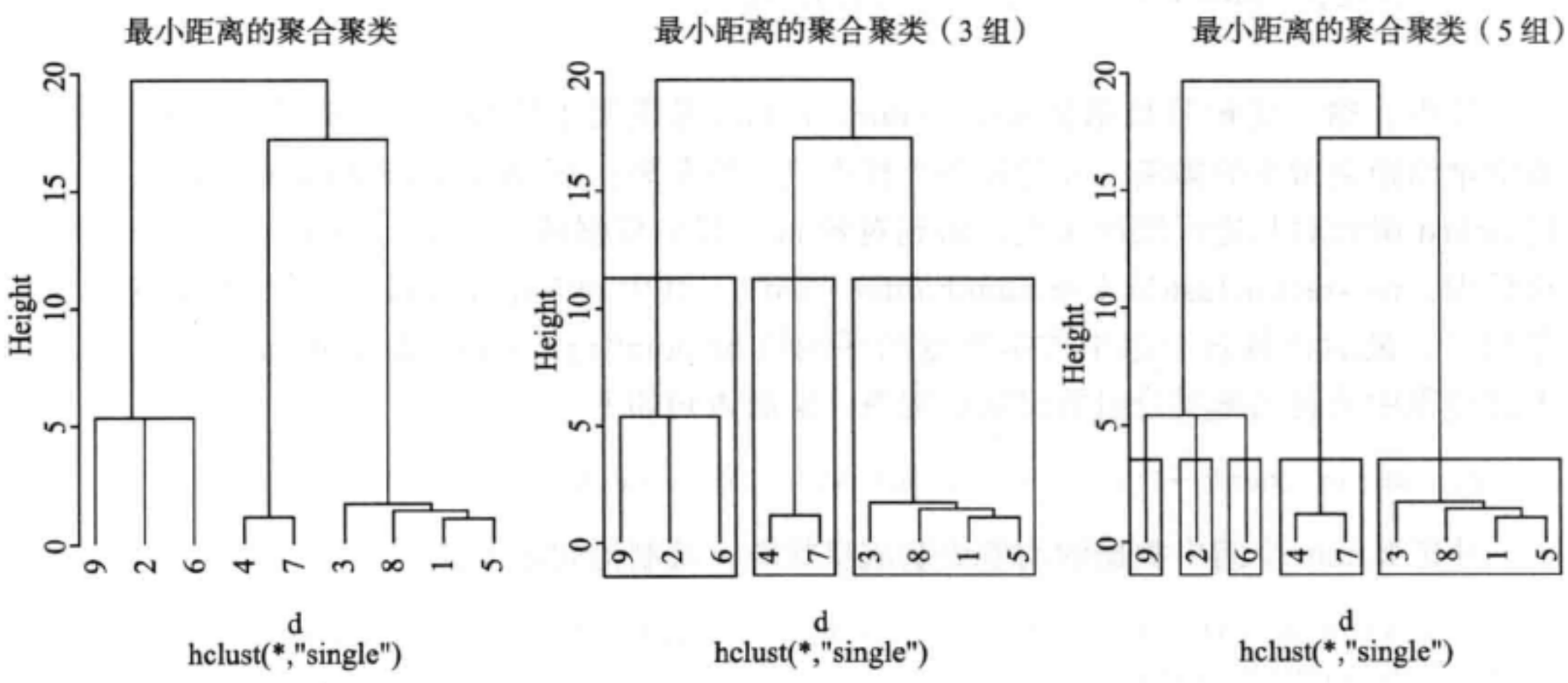


图 8-9 最小距离聚合示意图

其中 plclust() 函数用于绘制由 hclust() 函数得到的对象，rect.hclust () 函数用于把由 hclust() 函数得到的对象按照指定的数目进行聚类分组，并绘制在图中（红线区域内的样本点是一组）。

自上而下（最大距离分解法）：与自下而上（聚合）法相反，它使用最大距离来衡量一个组是否要进行分解，并对其进行不断分解，直到每个聚类均只包含一个样本为止。这里仍使用 hclust 函数，并设置 method 参数为 "complete" 来完成最大距离分解法。

2. 实例：问卷调查的样本聚类分析

下面使用层次聚类法对 8.1.4 节中问卷调查的因子分析结果进行聚类分析，首先自定义一个层次聚类分析函数 hc()。代码如下：

```
hc<-function(data_frame,k.num){
    d<-dist(scale(data_frame))
    hc<-hclust(d)
    dev.new()
    plclust(hc,hang=-1)
    re<-rect.hclust(hc,k=k.num,border="red")
    # re 划分 5 个聚类，re[[i]] 是第 i 个聚类包含的样本 id 向量
    dev.new()
    hc.point<-data_frame[1:k.num,]
    # hc.point 用于存储每个聚类里各变量的平均值
    for(i in 1:k.num){
        for(j in 1:length(names(data_frame))){
            hc.point[i,j]<-mean(data_frame[re[[i]],j])
        }
    }
     stars(hc.point+1+round(abs(min(hc.point))),full=F,draw.segments=T,key.
loc=c(5,0.5),mar=c(2,0,0,0))
    list(hc.mean=hc.point,hc=hc,re.hc=re)
}
```

其中，输入变量是数据框 data_frame，k.num 是聚类组的数目。首先使用 scale 函数消除纲量给距离带来的影响，并计算各个样本点间的距离：d<-dist(scale(data_frame))。然后使用 hclust 函数对其进行层次聚类，得到对象 hc。接着根据输入变量 k.num 对结果 hc 进行分组处理：re<-rect.hclust(hc,k=k.num,border="red")。其中 re[[i]] 表示第 i 组分组中对应的样本序列号。最后计算各个组中样本数据的平均值 hc.point[i,j]<-mean(data_frame[re[[i]],j])。在上述过程中绘制出聚类分组的树状结构图，绘制语句如下：

```
plclust(hc,hang=-1); re<-rect.hclust(hc,k=k.num,border="red");
```

并把 k.num 个组中数据的均值绘制成星状图，绘制语句如下：

```
stars(hc.point+1+round(abs(min(hc.point))),full=F,draw.segments=T, key.
loc=c(5,0.5),mar=c(2,0,0,0))
```

下面将 8.1.4 节中得到的因子分析模型 sol.factor 中的因子得分数据 sol.factor$score 作为

输入内容，进行层次聚类分析。代码如下：

```
data<-as.data.frame(sol.factor$score)
sol.hc<-hc(data,k.num=4)
```

结果如图 8-10 和图 8-11 所示。

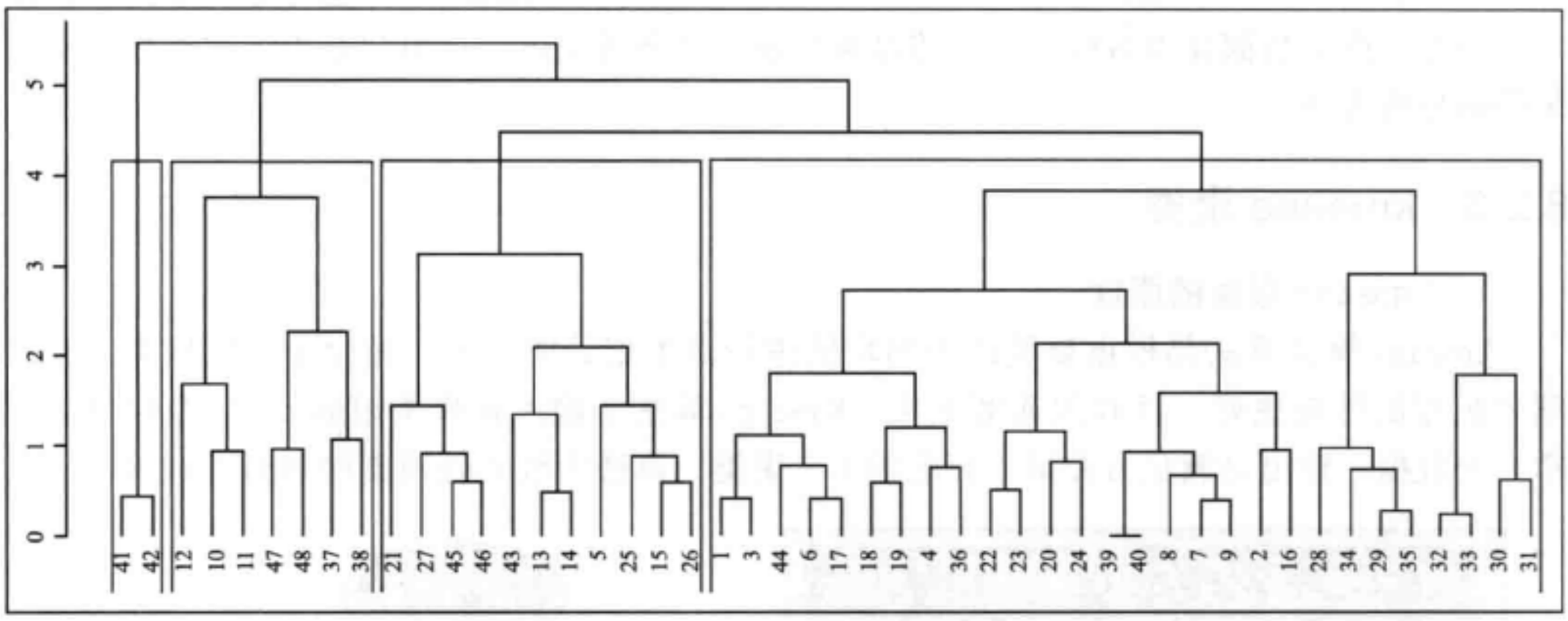


图 8-10 聚类分组的树状结构图

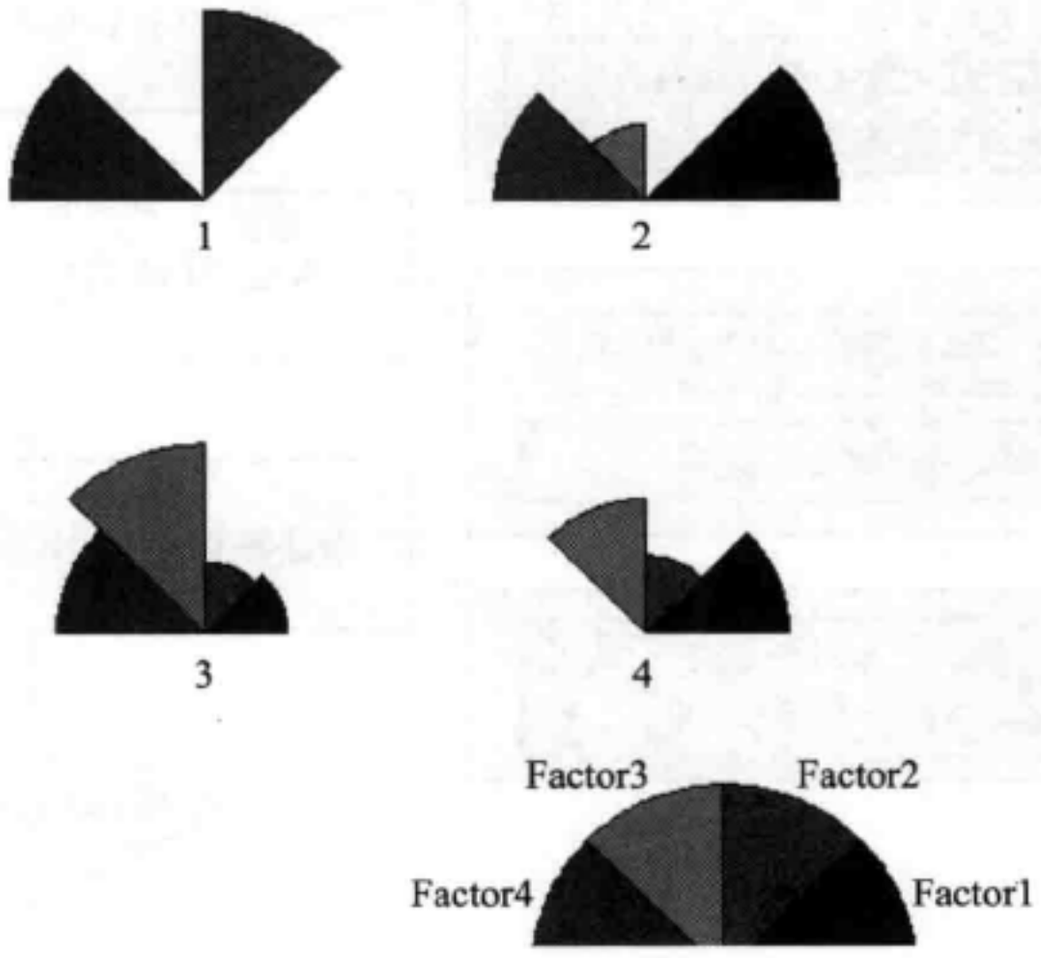


图 8-11 各组中数据均值的星状图

根据以上两个输出图，可以得到如表 8-21 所示的分析结果。

表 8-21 分析结果

组序号	组特征	用户占比
1	因子 2 和因子 4 得分极高，该类用户非常关注商品样式和质量	2/48=4.16%

（续）

组序号	组特征	用户占比
2	因子 1 和因子 4 得分很高，该类用户比较关注商品价格和质量	7/48=14.58%
3	因子 3 和因子 4 得分较高，该类用户比较关注网站服务和商品质量	11/48=22.91%
4	因子 1 和因子 3 得分较高，该类用户比较关注商品价格和网站服务	28/48=58.33%

当然，由于数据样本数较少，上述结果可能不具备可信度，这里只是以该数据作为实例来说明分析方法。

8.2.3 kmeans 聚类

1. kmeans 聚类的原理

kmeans 聚类算法是根据聚类组中的均值进行聚类划分的方法，相比于层次聚类，其处理大数据的性能更好，且收敛速度更快。Kmeans 算法的输入是聚类组数 k，以及包含 n 个样本的数据。输出是满足方差最小标准的 k 个聚类，算法大致的处理流程如图 8-12 所示。

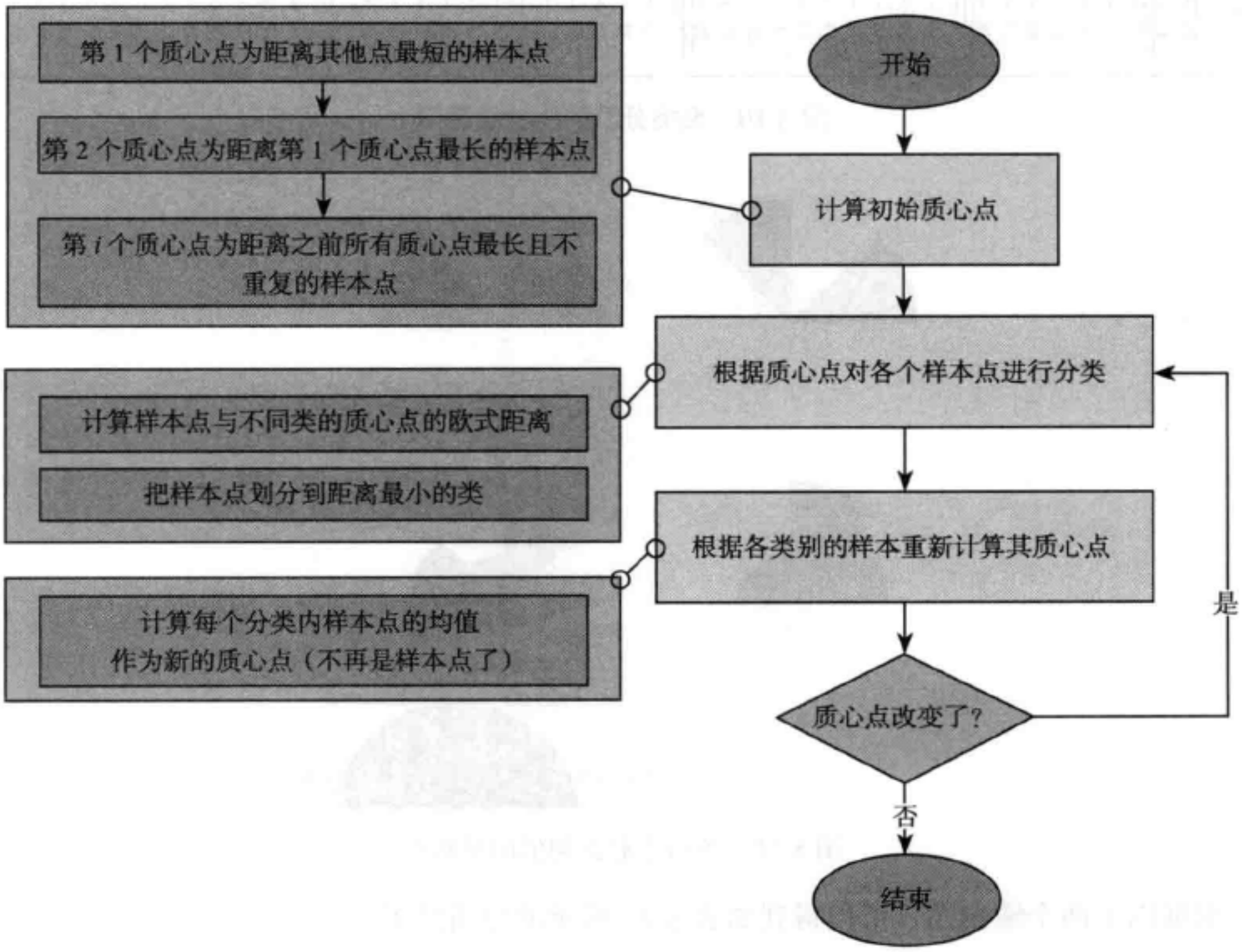


图 8-12 kmeans 算法处理流程

（1）确定初始质心点

在确定初始质心点时，先把与其他样本点距离最小的样本点作为第一个初始质心点。然

后将距离第一个初始质心点最远的样本点作为第二个初始质心点。再把距离第 1 个和第 2 个初始质心点的平均距离最远的样本点作为第 3 个初始质心点。以此类推，即可把距离所有 i-1 个初始质心点的平均距离最远的样本点作为第 i 个初始质心点。直到计算好所有 k 个初始质心点为止。

（2）根据质心点对各个样本进行分类

依次计算各个样本点与质心点的距离，并把其归于最近一组质心点的组别中，这里假设要把 n 个样本数据（包括 3 个变量）聚类到 4 个组中，如表 8-22 所示。

表 8-22 分类示例

样本号	样本坐标	质心点 1 (1,3,2)	质心点 2 (5,13,2)	质心点 3 (5,13,22)	质心点 4 (10,-13,-29)	样本归类的组别
1	(1,1,1)	2.236068	12.68858	24.5153	34.30743	1
2	(2,3,10)	8.062258	13.15295	15.90597	42.90688	1
…	…	…	…	…	…	
n	(4,11,31)	30.23243	29.08608	9.273618	64.89992	3

在表 8-22 中，依次计算样本坐标数据与各个质心点坐标数据的欧式距离。可以看到，样本点 1 与质心点 1 的距离最近，于是把样本点 1 归到第 1 组中。同理，样本点 2 也会归到第 1 组中，最后一个样本点 n 则归到第 3 组中。

（3）重新计算各个分类内的质心点

以分好的组别中各个样本点的均值点作为新的质心点，如表 8-23 所示。

表 8-23 质心点示例

组别	样本数据					均值质心点数据
1	(1,1,1)	(2,3,10)	(1,4,11)	(2,4,9)	(1,1,9)	(1.4,2.6,8)
2	(10,17,21)	(12,13,30)				(11,15,25.5)
3	(3,15,28)	(4,10,30)	(5,12,30)	(4,11,31)		(4,12,29.75)
4	(12,−17,−21)	(10,−12,−27)				(11，−14.5，−24)

在表 8-23 中计算出各个组别内样本点数据的均值，并以此作为新的质心点数据。同时判定新的质心点是否等于上一次的质心点，如果等于，则说明质心点不变，此时结束算法计算，否则跳至步骤（2）继续进行 kmeans 算法分析。

2. R 语言实现

在 R 语言中，使用 kmeans 函数来进行 kmeans 算法计算，其参数说明如表 8-24 所示。

表 8-24 kmeans 函数参数说明

参数	说明
x	样本数据
centers	聚类的组数
iter.max	最大迭代次数，默认为 10

下面的代码产生了包括两个变量的 220 个样本数据，并绘制于二维平面内：

```
    x<-c(rnorm(60,5,2),rnorm(20,7,2),rnorm(30,39,2),rnorm(40,21,2),rnorm(30,-
19,2),rnorm(40,-1,2))
    y<-c(rnorm(60,5,2),rnorm(20,7,2),rnorm(30,39,2),rnorm(40,21,2),rnorm(30,-
```

```
19,2),rnorm(40,-1,2))
    data<-data.frame(x=x,y=y)
    plot(data,pch=16 ,cex=2)
```

绘制结果如图 8-13 所示。

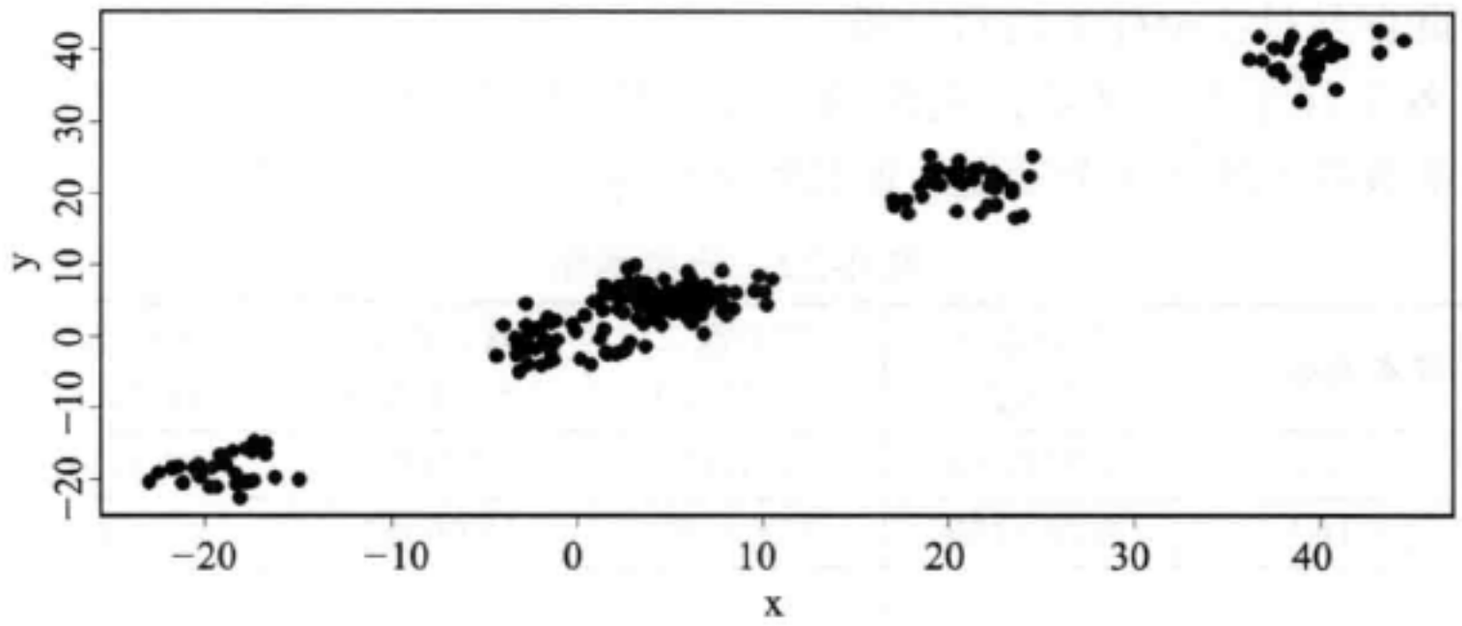


图 8-13 结果展示图

可见样本数据大致可以聚类为 4 ～ 5 个组别。

然后使用 kmeans() 函数对其进行聚类，并把相同组别的样本点绘制为同一颜色。代码如下：

```
plot.kmeans<-function(data,cluster.id,center,main.lab=""){
    centers<-length(unique(cluster.id))
    col<-rainbow(n=centers,start=0,end=5/6)
    plot(data,pch=16,col=col[cluster.id],cex=2,main=main.lab)
    for(i in 1:centers){
        points(center[i,1],center[i,2],pch=17,col=col[i],cex=2.5)
        text(center[i,1],center[i,2],labels=paste("第",i,"个中心"),col="black")
    }
}
par(mfcol=c(1,3))
sol4<-kmeans(data,4)
plot.kmeans(data,sol4$cluster,sol4$centers,"kmeans 聚类为 4 组")
sol5<-kmeans(data,5)
plot.kmeans(data,sol5$cluster,sol5$centers,"kmeans 聚类为 5 组")
sol6<-kmeans(data,6)
plot.kmeans(data,sol6$cluster,sol6$centers,"kmeans 聚类为 6 组")
```

绘制结果如图 8-14 所示。

注意，由 R 语言的 kmeans 函数得到的结果每次并不相同，这主要是由于初始质心点计算会有部分随机性。

最后通过计算各组别内样本点与最后质心点的距离平方和来衡量聚类的性能。代码如下：

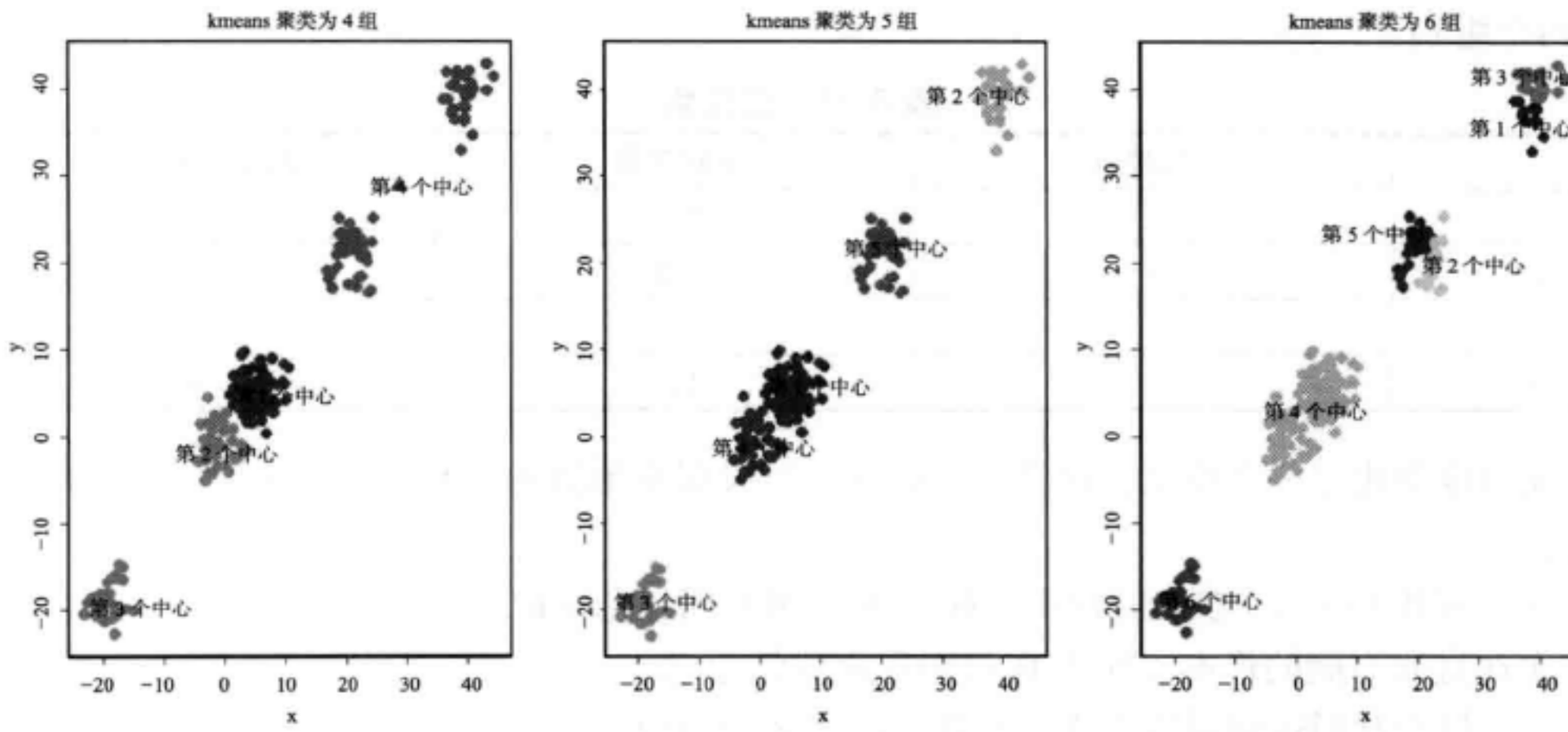


图 8-14 聚类组数分别取 4 ～ 6 时的示意图

```
kmeans.test<-function(data,cluster.id,center){
    result<-0
    for(i in 1:nrow(data)){
        result<-result+(data[i,1]-center[cluster.id[i],1])^2+
                (data[i,2]-center[cluster.id[i],2])^2
    }
    result
}
kmeans.test(data,sol4$cluster,sol4$centers)
    [1] 4031.788
kmeans.test(data,sol5$cluster,sol5$centers)
    [1] 1915.446
kmeans.test(data,sol6$cluster,sol6$centers)
    [1] 3783.385
```

可见，在对 5 聚类分组后，其组别内的样本点与最后质心点的距离平方和是最小的（为 1915.446），因此可以使用 5 分组对其进行 kmeans 聚类。

kmeans 也可以用于 8.1.4 节中的问卷调查结果。代码如下：

```
data<- scale(as.data.frame(sol.factor$score))
sol<-kmeans(data,5)
```

此外，笔者自己也编写了一个 kmeans 代码，感兴趣的读者可以在 http://www.bassary.com/?p=1421 下载阅读。

8.3 样本判别

判别分析是用于判别个别样本所属的群体。例如，表 8-25 中共有 200 个样本，且分成

AB 两个组别。

表 8-25 数据集

	x 轴变量	y 轴变量	所属组别 group
1	1	20	A
2	30	−20	B
…	…	…	…
200	9	12	A

我们希望建立一个模型，使得可以通过一组规则来预测新样本所属的组别。假设判别规则是：

- 在直线 $y=x-20$ 左侧的样本点属于 A 组别并用红色标记。
- 在直线右侧的样本点属于 B 组别并用蓝色标记。

然后根据判别规则对样本进行绘制，如图 8-15 所示。

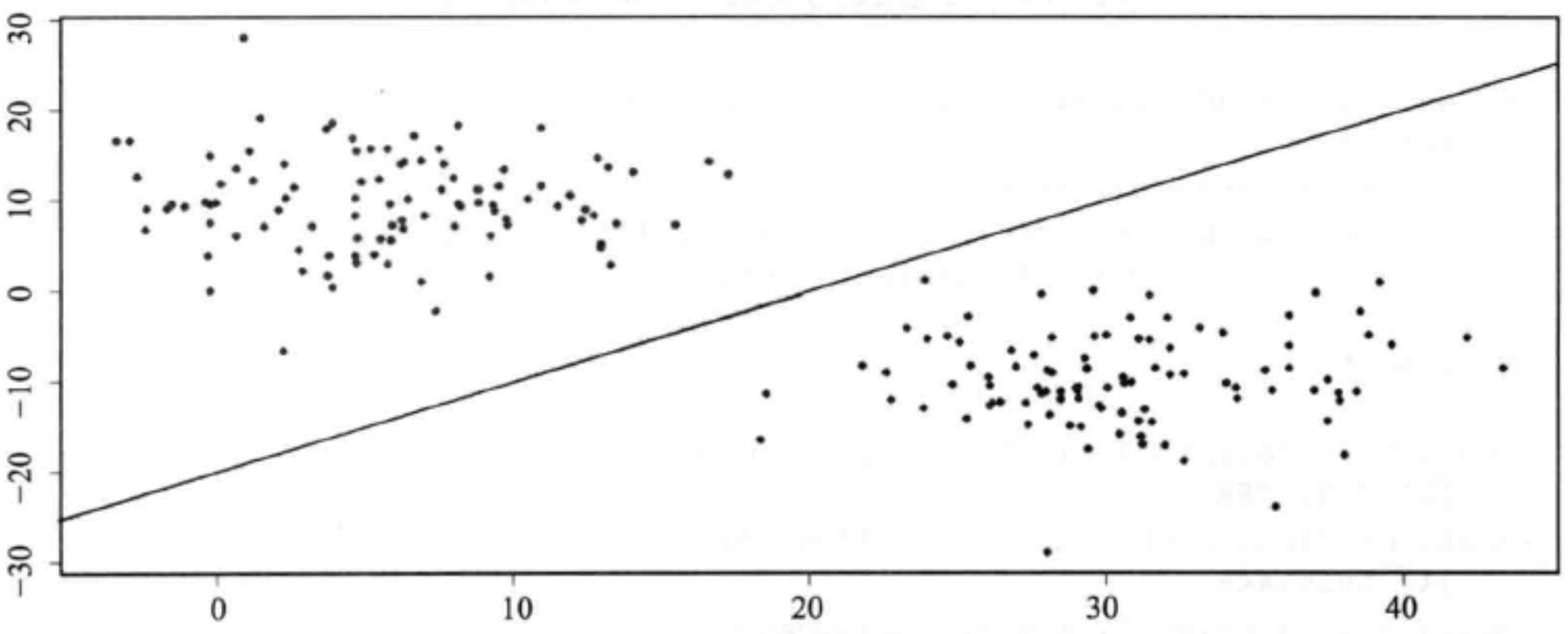


图 8-15 根据样本规则绘制的图

读者可能会突然意识到，这不就是第 7 章的分类分析问题吗？即通过 x 轴变量和 y 轴变量来预测组别变量：

$$\text{group}=f(x, y)$$

没错，如果可以单纯地像上面的假设一样使用一组简单的规则（一条直线）建立出 $\text{group}=f(x, y)$ 模型，这确实就是分类分析。然而，如果无法找到一套合适的规则（分类模型），那该怎么办呢？

此时要做如下处理：首先计算与新样本（x_{new}, y_{new}）距离最近的 9 个样本，然后查看其所属的组别。如果多数样本属于 A 组，则新样本（x_{new}, y_{new}）就预测属于 A 组，反之属于 B 组。上述方法的实质是 knn（k 最近邻分类）算法，它在只使用旧的样本数据而不建立任何规则（分类模型）的情况下，就可以预测新样本数据的所属组别，这套方法即为判别分析。

判别分析的典型应用是推荐系统的建立，本章以 knn（k 最近邻分类）算法为例，讲解如何建立推荐算法。

8.3.1 knn（k 最近邻分类）算法

1. 原理

假设训练数据中，每个样本点都含有变量 y，那么 y 值刚好就是该样本点的所属组别。例如，如果有两个类，那么 y 就是一个二元的变量。Knn（k- 最近相邻）方法指的是在训练数据集中动态确定与新样本点（$case_{new}$）距离最相近（根据自变量 Xi 来计算距离）的 k 个样本点 ($case_1$, $case_2$, .., $case_k$)，并利用这 k 个样本点的所属组别（$y_1,y_2,..,y_k$）来确定新样本点所属组别 (y_{new}) 的方法，公式如下：

$$y_{new}=\text{mode}\ (y_1,y_2,..,y_k)$$

即 y_{new} 等于 $y_1, y_2,\cdots,y_k$ 序列众数（mode）的取值。

注意对参数 k 的确定：

- 如果 k=1，即 1-NN，则新样本点的所属组别 y_{new} 就是最近样本的所属组别 y_1。可以证明，如果有大量的数据及充分复杂的分类规则，则该方法在减少误差率方面，最多能减到用 1-NN 算法规则时的一半。
- 如果增加 k 值，则可以提供一个平滑的分类，以便减少训练数据中噪声样本的影响。在典型的应用中，k 的取值为个位数或十位数，而不是百位或千位数。
- 如果 k 值很高并接近于训练集样本数，则仅预测在训练数据集中大多数样本所属的组别，而不用管新样本的值如何，这是一个过度平滑的例子。

2. Knn 函数实现

在 R 语言中，使用 class 包的 knn 函数可以实现 knn 算法的基本功能，该函数的具体参数如表 8-26 所示。

表 8-26 knn 函数的参数说明

参数	说明
train	训练集中的自变量 Xi，相当于 8.3.1 节中样本 case 中的自变量 Xi 数据
test	测试集中的自变量 Xi，相当于 8.3.1 节中新样本 $case_{new}$ 中的自变量 Xi 数据
cl	训练集中的组别数据，相当于 8.3.1 节中样本 case 的组别数据 y
k	k 参数取值
prob	是否计算预测组别的概率，默认为 FALSE，不计算。 如果为 TRUE，则结果中的 prob 属性可以存放该概率的数值

例如，要对表 8-27 中的数据进行 knn 算法处理，预测测试集中的样本组别。

表 8-27 预测样本组别

	训练集 train 和 cl		测试样本 1(8,7,15) 与训练集的距离
1	11,9,3	A	12.52996
2	7,0,31	B	17.49286
3	9,7,4	A	11.04536
4	5,6,8	A	7.681146
5	6,−1,20	B	9.643651
预测 (k=3) 组别			距离最近的 3 个样本是 3(A)、4(A)、5(B)，预测组别是 A
预测概率 prob			2/3=0.66667

使用R语言实现的代码如下：

```
library('class')
train<-matrix(c(11,7,9,5,6,9,0,7,6,-1,3,31,4,8,20),ncol=3);train
     [,1] [,2] [,3]
[1,]   11    9    3
[2,]    7    0   31
[3,]    9    7    4
[4,]    5    6    8
[5,]    6   -1   20
cl<-as.factor(c("A","B","A","A","B"))
test<-matrix(c(8,7,15),ncol=3)
sol.knn<-knn(train,test,cl,k=3,prob=TRUE);sol.knn
[1] A
attr(,"prob")
[1] 0.6666667
Levels: A B
```

查看最后的结果，了解到组别是A，概率（可以使用attr(sol.knn, "prob")直接读取）是0.666667。

8.3.2 实例：基于knn算法的商品推荐系统

下面基于knn算法建立一个物品相关性协同过滤推荐系统。

1. 训练集数据train.data

以表8-28所示的某视频网站的用户收藏行为数据作为建立推荐系统的训练集数据。

表8-28 数据集

	userid	moviesid	ratings
1	1	1029	1
2	1	1907	1
…	…	…	…
127787	6037	1945	1
127788	6040	593	1

其中userid是用户ID，共6023个用户，userid并不是顺序排列的。moviesid是视频编号，共498部视频。ratings是用户收藏行为标识，这里的取值均为1，表示该用户收藏了该视频。第一行和第二行分别表示用户1（userid=1）收藏了视频1029和1907（Moviesid=1029和1907）。对于其他未收藏的视频并不在数据集中展示。注意，对于是否收藏这样的二元变量，ratings实际无过多意义（训练集中取值恒为1），因为ratings取0时的数据并未在数据集中展示。但是如果rating代表用户投票数值，则很有意义。

下面把上述数据存放于train.data数据框中，其前3行的样本数据如下：

```
train.data[1:3,]
      userid  moviesid   ratings
1        1      1029         1
2        1      1907         1
3        1      3105         1
```

2. 建立用户—物品矩阵 user.item.matrix

现在要把常规的 train.data 形式的矩阵转化为列代表不同视频、行代表不同用户、内部元素代表是否收藏的矩阵，即每列表示某一视频被哪些用户所收藏，每行表示某一用户收藏了哪些视频，如表 8-29 所示。

表 8-29 收藏矩阵

		共 498 列，每列代表一个视频，列名称是 Moviesid						
		16	26	28	29	…	3936	3944
共 6023 行，每行代表一个用户，行名称是 userid	1	0	0	0	0	…	0	0
	2	0	0	0	0	…	0	0
	3	0	0	0	0	…	0	0
	4	0	0	0	0	…	0	0
	5	1	0	0	1	…	0	0
	…	…	…	…	…	…	…	…
	6040	0	0	0	1	…	0	0

例如，第 5 行第 1 列的元素为 1，代表用户 5（行名称为 5）收藏了视频 16（列名称是 16）。代码如下：

```
library('reshape')
user.item.matrix<- cast(train.data, userid ~ moviesid, value = 'ratings',fill=0)
row.names(user.item.matrix) <- user.item.matrix[,1]
user.item.matrix<-user.item.matrix[,-1]
```

这里的核心函数是 reshape 包的 cast 函数，如下代码执行后会返回行代表的用户。

```
user.item.matrix<- cast(train.data, userid ~ moviesid, value = 'ratings',fill=0)
```

其中，列代表视频（userid ~ moviesid），元素内容是 ratings (value = 'ratings') 的矩阵 user.item.matrix，默认值 NA 自动使用 0 代替（fill=0），以代表用户未收藏视频。同时 user.item.matrix 的列名称为视频编号 moviesid，首列数据是每行所代表用户的 userid。

然后使用如下代码把每行的名称直接设置为 userid，并删除首列数据。

```
row.names(user.item.matrix) <- user.item.matrix[,1]
user.item.matrix<-user.item.matrix[,-1]
```

3. 计算物品距离矩阵 distance

这里使用视频间的相关性系数来间接计算视频的距离矩阵。例如，user.item.matrix 矩阵中的第 1 列和第 2 列分别代表视频 16 和视频 26 被不同用户收藏的情况，其相关性系数为：

```
sim_cor<-cor(user.item.matrix[,1], user.item.matrix[,2]); sim_cor
[1]  0.08902468
```

sim_cor 的取值范围是 [−1，1]，这里使用如下规则把 sim_cor 转化为 distances。

- 当 sim_cor=−1 时，distances=inf 无限大。
- 当 sim_cor=1 时，distances=0。

代码如下：

```
distance<-  -log((sim_cor/2) + 0.5); distance
[1]  0.6078647
```

最后得到视频 16 和视频 26 的距离是 0.6078647。如果要计算每个视频间的距离，则只需使用如下代码修改 cor 函数内的相应参数即可。

```
sim_cor<-cor(user.item.matrix)
distance<- -log((sim_cor/2) + 0.5)
```

其中，各行和各列均表示一个视频。例如，1 行 2 列和 2 行 1 列的元素均表示上例中视频 16 和视频 26 的距离。

```
distance[1,2]
[1]   0.6078647
distance[2,1]
[1]   0.6078647
```

最后把 distance 矩阵的行名称和列名称设置为视频编号 moviesid，以便可以在后续程序的 distance 矩阵中定位出实际的视频数据。

```
colnames(distance)<-colnames(user.item.matrix)
rownames(distance)<-colnames(user.item.matrix)
```

4. 计算用户和物品的关系程度

自定义 knn.userid.item 函数是算法的核心，它主要计算用户 userid 和物品（视频）itemid 的相关程度。通过 distance 物品间的距离矩阵（图 8-16 中的表 1），可以形象地假设得到一个商品二维平面分布图（图 8-16 中的图 1）。

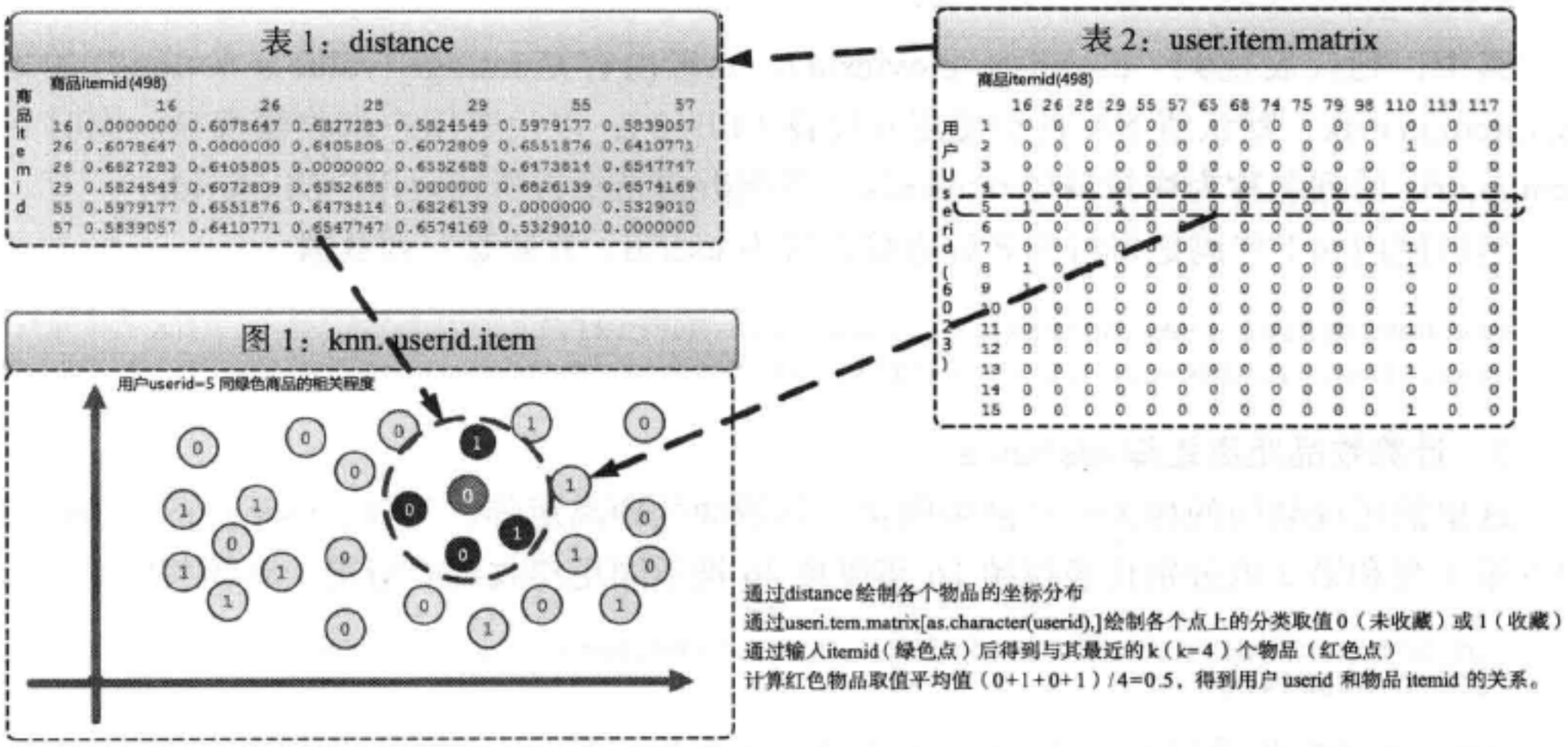


图 8-16 示意图

其中各个物品的分布位置由 distance（图 8-16 中的表 1）得出。各点上的分类取

值由用户—物品矩阵 user.item.matrix 中行名称是 userid 的行数据 user.item.matrix [as.character(userid),] 得到，分类取值 0 表示未收藏，1 表示收藏。然后通过 knn 算法思想，找到与商品 itemid（绿色点表示）距离最近的 k 个物品（红色点表示），这里 k 的值为 4。最后计算红色物品的分类取值之和（与 itemid 距离最近的 k 个物品中，被用户 userid 所收藏的物品数），并除以 k，从而代表用户 userid 和物品 itemid 的关系。

在实际代码实现中，不必计算各个物品在图 8-16 中图 1 的坐标（因为无法计算），但是可以找到与 itemid 距离（关系）最近的 k 个物品。代码如下：

```
knn.userid.item<-function(userid,itemid,user.item.matrix,distance,k=25){
    item.index<-which(rownames(distance)==as.character(itemid))
    k.nearest.item.index<-order(distance[item.index,])[2:(k + 1)]
    k.nearest.itemid<-as.numeric(rownames(distance)[k.nearest.item.index])
    sum(user.item.matrix[as.character(userid),as.character(k.nearest.itemid)])/k
}
```

下面通过 distance 物品距离矩阵找到与 itemid 距离最近的 k 个物品。

1）查找名称是 itemid 的物品在 distance 的序列号 item.index。代码如下：

```
item.index<-which(rownames(distance)==as.character(itemid))
```

2）如果 order（< 向量 a>）可以返回向量 a 的取值由小到大排序后元素的序列号信息，则语句如下：

```
k.nearest.item.index<-order(distance[item.index,])[2:(k + 1)]
```

上述语句可以返回 distance[item.index,] 中值最小的第 2 至 k+1 元素的序列号。k.nearest.item.index。注意，由于 distance 中与 itemid 距离最近的物品还是 itemid，所以这里取第 2 至第 k+1 个元素，而非第 1 至第 k 个元素。

3）使用 k.nearest.item.index 反查出物品 IDk.nearest.itemid。代码如下：

```
k.nearest.itemid<-as.numeric(rownames(distance)[k.nearest.item.index])
```

注意：由于这里已经得到了物品间的距离矩阵 distance，所以不必再使用 knn 函数进行计算，而是依据 knn 算法的含义（距离最近的 k 个样本数据）来自定义代码进行计算。

下面使用用户—物品矩阵 user.item.matrix 来计算距离最近的 k 个物品中有多少个物品被 userid 收藏，再除以 k 得到一个 [0,1] 的数据，用以表明用户 userid 和物品 itemid 的关系。语句如下：

```
sum(user.item.matrix[as.character(userid),as.character(k.nearest.itemid)])/k
```

5. 计算用户的推荐视频

使用自定义的核心函数 knn.userid.item，可以计算用户 userid 与所有商品的关系数据，并可取关系最大的视频作为推进视频。该过程实际就是在上节中的物品分布图中，查找周围物品（与之最近的 k 个物品）中取值为 1（表示收藏）最多的物品。

例如，在图 8-17 中，绿色物品周围（蓝色虚圈内）的物品的分类取值均是 1，所以它会被作为一款推荐物品。

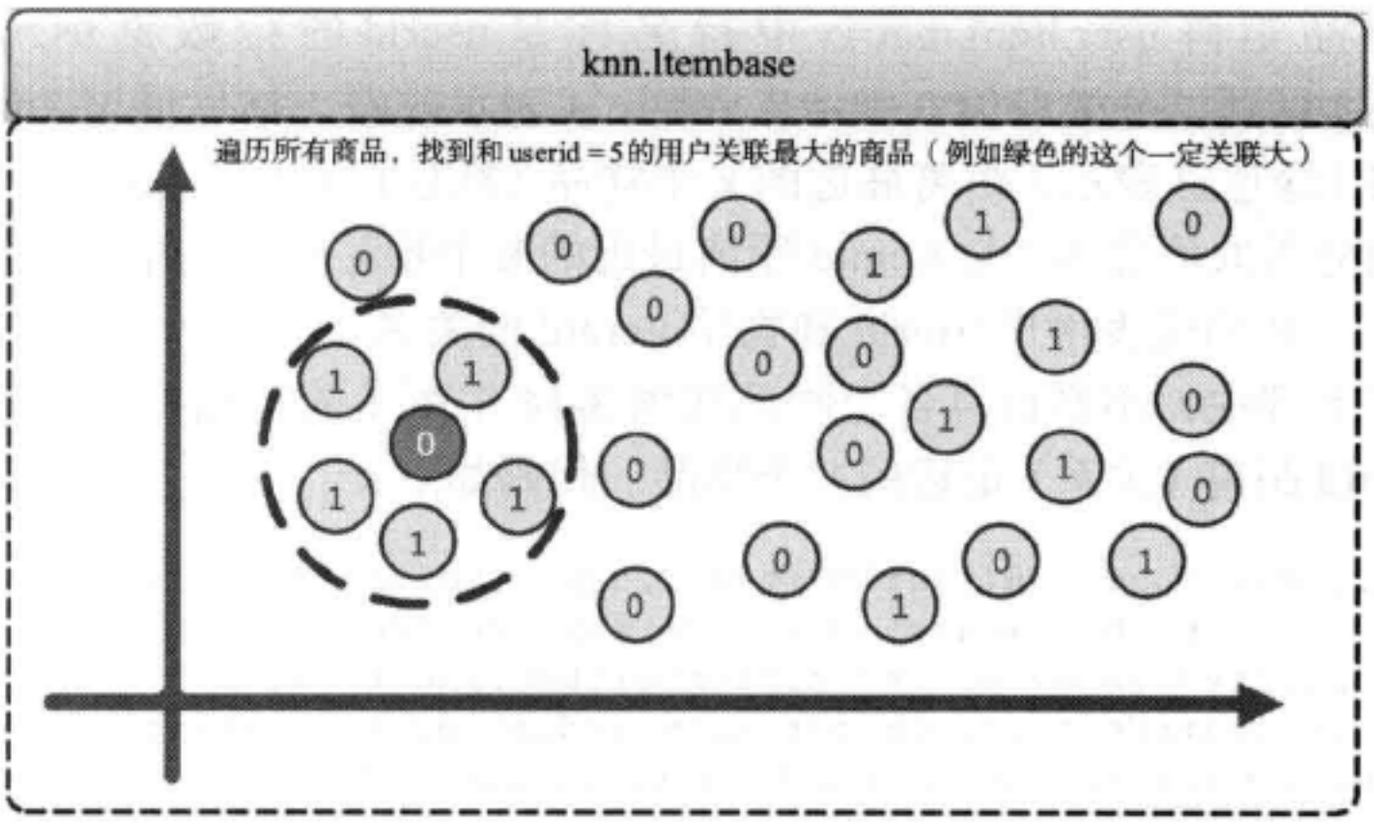


图 8-17 查找周围物品中取值为 1 最多的物品示意图

该过程涉及的具体代码如下：

```
    knn.itembase<-function(userid,user.item.matrix,distance,k=25,return.item.
num=10){
        knn.userid<-0
        for(i in 1:nrow(distance)){
          knn.userid[i]<-knn.userid.item(userid,rownames(distance)[i],
                                         user.item.matrix,distance,k=k)
        }
        return.itemid<-rownames(distance)[order(knn.userid,decreasing = TRUE)]
        return.itemid<-setdiff(return.itemid,
            colnames(user.item.matrix)[which(user.item.matrix[as.character
            (userid),]!=1)])
        return.itemid[1:return.item.num]
    }
```

代码中的 knn.userid 用于存储用户 userid 和所有物品的关系数据，然后依据 knn.userid 取值由大到小进行排序，返回其物品编号 itemid。代码如下：

```
return.itemid<-rownames(distance)[order(knn.userid,decreasing = TRUE)]
```

然后使用如下语句找到已经被用户收藏的物品的 itemid。

```
colnames(user.item.matrix)[which(user.item.matrix[as.character(userid),]!=1)])
```

再从上面得到的 return.itemid 中提出已收藏物品的信息。代码如下：

```
return.itemid<-setdiff(return.itemid,
colnames(user.item.matrix)[which(user.item.matrix[as.character(userid),]!=1)])
```

最后依据推荐商品数来返回最后为用户 userid 推荐的 return.item.num 个物品 itemid。例如，要为 userid 是 4169 的用户推荐 10 个视频，knn 的参数 k 为 25，代码如下：

```
knn.itembase(4169,user.item.matrix,distance,k=25,return.item.num=10)
[1]" 820" " 1351"" 1384"" 2534"" 3781"" 210" " 3492"" 320" " 458" " 991"
```

第9章
样本间的关系

9.1 关联分析

关联分析主要用于从数据集中发现数据项（item）之间的联系，最著名的应用实例是沃尔玛的啤酒尿布推荐销售案例。20 世纪 90 年代，美国沃尔玛超市的管理人员分析销售数据时，发现了一个令人难以理解的现象：在某些特定的情况下，“啤酒”与“尿布”两件看上去毫无关系的商品会经常出现在同一个购物篮中，经过后续调查发现，这种现象出现在年轻的父亲身上。在美国有婴儿的家庭中，一般是母亲在家中照看婴儿，年轻的父亲前去超市购买尿布。父亲在购买尿布的同时，往往会顺便为自己购买啤酒，这样就会出现啤酒与尿布这两件看上去并不相干的商品会经常出现在同一个购物篮的现象。在发现这一独特的现象后，沃尔玛超市开始尝试将啤酒与尿布摆放在相同的区域，让年轻的父亲可以同时找到这两件商品，并很快地完成购物。而沃尔玛超市也可以让这些客户一次购买两件商品而不是一件，从而获得了很好的商品销售收入。

9.1.1 关联分析的原理

如果将商场销售的所有商品设为一个集合，那么每个商品（item）均为一个二元的分类变量（取值为真 / 假），用以描述相应商品是否被某顾客（某一个顾客在某一次）购买。因此每个顾客的购物袋就可以用一个布尔向量来表示。分析相应布尔向量就可获得哪些商品是被一起购买（关联）的，从而了解用户的购物模式。表 9-1 是 4 个商品（item1 ～ item4）的 7 次购买数据，分别用 basket1 ～ basket7 表示。

表 9-1 数据集

	Item1	Item2	Item3	Item4		Item1	Item2	Item3	Item4
Basket1	1	0	1	0	Basket5	0	0	1	0
Basket2	0	0	1	1	Basket6	1	0	1	1

（续）

	Item1	Item2	Item3	Item4		Item1	Item2	Item3	Item4
Basket3	1	1	1	1	Basket7	0	1	1	1
Basket4	1	1	0	0					

1. 信任度

由表 9-1 可以观察到，顾客在购买商品 item4 时 100% 也会购买商品 item3，于是可以说“item4 和 item3 的关联性很大”，可使用信任度（confidence）来衡量。如果把购买商品 item4 记为 A，购买商品 item3 记为 B，则 item4=>item3 的信任度等于同时购买 item4 和 item3 的次数除以购买 item4 的次数：

```
confidence(item4=>item3)=confidence(A=>B)=P{B|A}= P{AB}/P{A}=4/4=100%
```

2. 支持度

如果在集合中，item3 和 item4 虽常被同时购买，但在整个集合中购买 item3 和 item4 的情况却较少，则由信任度得到的任何结论均不可靠，这时可以使用支持度（support）来衡量同时满足 A 和 B 的概率。item4=>item3 的支持度等于同时购买 item4 和 item3 的次数除以集合总数：

```
support(item4=>item3)=support(A=>B)=P{AB} =4/7= 57.14%
```

3. 提升度

从表 9-1 可计算出 item3 自身的支持度是 support(item3)=P{item3}=6/7=85.71%，即不做任何推荐时，用户购买 item3 的概率是 85.71%。而由 confidence(item4=>item3)=100% 可知，如果用户购买了商品 item4 后再为其推荐商品 item3，则用户购买 item3 的概率为 100%。可见购买 item3 的概率提高了 (100%)/(85.71%)=1.667。我们把这两次概率的比值称为提升度（lift），用于衡量关联规则是否可用：

```
lift(item4=>item3)= lift(A=>B)= confidence(A=>B)/ support(B)= (100%)/(85.71%)=1.667
```

如果提升度大于 1，则表示推荐（关联）商品的购买概率比未推荐前有所提高，即该条关联规则有用。反之表示推荐商品的购买概率还不如推荐前的概率，即该条关联规则无用。

综上所述，可以说：如果一个顾客购买了商品 item4，则有 100%（信任度）的可能性也会购买商品 item3。而这种情况（即购买了商品 item3，也购买了商品 item4）发生的可能性是 57.14%（支持度）。通过商品 item4 而为用户推荐 item3 的购买概率是单纯购买 item3 概率的 1.667 倍（提升度）。

4. 关联分析

如果设置取值范围为 [0%,100%] 的最小支持度阈值 min_sup 和最小信任度阈值 min_confid，则关联分析的实质就是在集合（itemset）中找到符合以下条件的关联规则 $item_i$=>$item_j$，并依据提升度 lift 找出其中可用的（lift>1）关联规则。

```
support(itemi=>itemj)> min_sup 且 confidence (itemi=>itemj)> min_confid
```

9.1.2 Apriori 算法的 R 语言实现

在关联分析中，由 Rakesh Agrawal 在 1994 年提出的 Apriori 算法是最为基础有效的算法。它使得支持度、信任度、提升度的计算更为高效，其原理不再详细讲解。在 R 语言中，Apriori 算法的核心函数是 arules 包的 apriori 函数。

1. 数据准备

（1）疏散矩阵方式

假设把表 9-1 的数据放于数据框 a 中，列是商品 item，行是每次购物的购物篮，并且如果商品在该购物篮中，则取 1，否则取 0（疏散矩阵），那么可得到如下语句：

```
library(arules)
    a<-matrix(0,7,4,dimnames=list(c("basket1","basket2","basket3","basket4","basket
5","basket6","basket7"),c("item1","item2","item3","item4")))
    a[1,]<-c(1,0,1,0)
    a[2,]<-c(0,0,1,1)
    a[3,]<-c(1,1,1,1)
    a[4,]<-c(1,1,0,0)
    a[5,]<-c(0,0,1,0)
    a[6,]<-c(1,0,1,1)
    a[7,]<-c(0,1,1,1)
```

可使用如下语句把上面得到的疏散矩阵 a 转化为 apriori 函数可用的数据结构。

```
a.class<-as(a,"transactions")
```

（2）数据框方式

有时候原始数据不适合转化为疏散矩阵，例如，图 9-1 所示的原始数据源自于微博留言。

	A	B	C	D	E	F	G	H	I	J	K
1	57995215	中小板	中签率	科伦药业	新高	中原特钢	破发	贵州百灵	资金量	机构参与	股票
2	58016574	天虹商场	市盈率	登陆中小板	价格区间	四新股	中国联通	运营商	商场	联通	市场份额
3	58016676	新股	股权激励	二级市场	股权激励产	价格	合康变频	公司	偏低	行权价	估值水平
4	58016862	新股	破发	期指套利	打新资金	股指期货	资金	收益	投资者	网下配售	打新股
5	58016932	宁波港	募集资金	南通江海	IPO申请	总股本	上交所	首发申请	深交所	计划	电容器
6	58026169	公司	科伦药业	募集资金	研发	项目	竞争优势	登陆中小板	品种	原材料	生产线
7	58026309	唐山港	港口	铁矿石	京唐港区	煤炭	钢铁	网络	今日招股	河北	吞吐量
8	58026461	上海锐奇	创业板IPO	首发申请	制造商	金属	证监会	上海	主要产品	快速发展	建筑
9	58066850	海峡股份	海南	客滚运输	航线	台海业务	泉州	客运	盈利模式	收购兼并	汽车
10	58066852	横店东磁	A股	股权激励计	交易日	发布公告	公司股票	均价	激励对象	草案	股权激励
11	58073363	持仓	全日	行情	合约	弱势	主力合约	主力	商品市场	弱势震荡	交易日

图 9-1 数据集实例

在图 9-1 中，每一行表示一个购物车内的 10 件商品（这里的购物车是一句微博留言，而商品是一些关键字，关联分析的目的则是分析出哪些关键字的关联性较大）。因为这里每行微博留言的关键字数均为 10，所以使用 read.csv 读取 csv 原文件后产生了一个 data.frame 数据框，其中第 1 列是原始数据中的行标号，无意义，将其去掉，最后的代码为：

```
data<-read.csv("D:/panguo/news_tags3.csv",header=F);data<-data[,-1]
data[1:2,]
    V2       V3     V4         V5       V6      V7       V8     V9     V10     V11
1  中小板   中签率 科伦药业   新高     中原特钢 破发     贵州百灵 资金量 机构参与 股票
2  天虹商场 市盈率 登陆中小板 价格区间 四新股   中国联通 运营商   商场   联通     市场份额
```

在 data 数据框中，使用行来表示不同商品在同一个购物车中。接下来将其转化为只有 2 列的数据框，其中第一列名为 bsk，用于表示商品的购物车 ID，第二列名为 item，用于表示真实的商品。例如，希望将原始数据中的第 1、第 2 行转化为如图 9-2 所示的格式。



	A	B	C	D	E	F	G	H	I	J	K
1	57995215	中小板	中签率	科伦药业	新高	中原特钢	破发	贵州百灵	资金量	机构参与	股票
2	58016574	天虹商场	市盈率	登陆中小板	价格区间	四新股	中国联通	运营商	商场	联通	市场份额
3	58016676	新股	股权激励	二级市场	股权激励产	价格	含康变频	公司	偏低	行权价	估值水平
4	58016862	新股	破发	期指套利	打新资金	股指期货	资金	收益	投资者	网下配售	打新股
5	58016932	宁波港	募集资金	南通江海	IPO申请	总股本	上交所	首发申请	深交所	计划	电容器
6	58026169	公司	科伦药业	募集资金	研发	项目	竞争优势	登陆中小板	品种	原材料	生产线
7	58026309	唐山港	港口	铁矿石	京唐港区	煤炭	钢铁	网络	今日招股	河北	吞吐量
8	58026461	上海锐奇	创业板IPO	首发申请	制造商	金属	证监会	上海	主要产品	快速发展	建筑
9	58066850	海峡股份	海南	客滚运输	航线	台海业务	泉州	客运	盈利模式	收购兼并	汽车
10	58066852	横店东磁	A股	股权激励计	交易日	发布公告	公司股票	均价	激励对象	草案	股权激励
11	58073363	持仓	全日	行情	合约	弱势	主力合约	主力	商品市场	弱势震荡	交易日
12	58081324	A股	估值水平	人民币升值	国内	个股	资本市场	市场竞争	国际	压力	风险
13	58081972	新股发行	解禁压力	个股	网上申购	申银万国	新股申购	限售股解禁	IPO	分析师	闽土股份
14	58082463	估值	地产	历史	价格	成交回暖	成交量	券商	行业	政策	房地产行业
15	58082473	钢铁	机会	钢价	中信证券	钢材价格	钢铁行业	钢铁业	行业	需求	钢企
16	58099511	三网融合	中国联通	焦点	主题	三条主线	中视传媒	每股收益	收入	补贴	央视
17	58115219	协议	两岸	双方	经济	委员会	服务贸易	合作机制	货物贸易	海峡	会议
18	58123819	业绩	公司	隆基机械	国内	生产线	效果	业绩增长	销售收入	OEM市场	结构调整
19	58124480	经济数据	美股	投资者	西班牙	美国经济	美国股市	欧元	稳定性	收低	流动性
20	58124482	经济数据	数据	欧股	低位	BP	英国	欧洲	美国	指数	股市

	L	M	N
		1	中小板
		1	中签率
		1	科伦药业
		1	新高
		1	中原特钢
		1	破发
		1	贵州百灵
		1	资金量
		1	机构参与
		1	股票
		2	天虹商场
		2	市盈率
		2	登陆中小板
		2	价格区间
		2	四新股
		2	中国联通
		2	运营商
		2	商场
		2	联通
		2	市场份额

图 9-2 转化示例

在图 9-2 中，第 m 列用于区分购物车，如果商品位于相同的购物车中，则第 m 列的编号相同。这样就可以使用一列来表示不同的购物车信息。实现该过程的代码如下：

```
bsk<-gl(nrow(data),ncol(data))
item<-as.vector(t(as.matrix(data)))
data_data.frame<-data.frame(TID=bsk,item=item)
```

注意，数据框不能直接转化为向量，需要先转化为矩阵再转化为向量。因为使用 as.vector 函数转化矩阵时，默认根据列来转化，所以要先把矩阵进行转置处理 t(as.matrix(data))。

最后使用 as（＜数据框对象＞,“ transactions”）将数据框转化为可以被 apriori 函数使用的对象。代码如下：

```
trans.list <- split(data_data.frame[,"item"], data_data.frame[,"TID"])
trans <- as(trans.list, "transactions")
```

这里使用 split 函数把具有相同 ID 号的 item 整理为一个字符串向量，用来作为列表对象 trans.list 的一个子元素，然后使用 as 函数进行数据格式转换。

2. 使用 apriori 函数进行关联分析

apriori 函数的常用参数如表 9-2 所示。

表 9-2 apriori 函数常用参数

参数	说明
data	集合数据，一般使用 as(…,"transactions") 进行转化
parameter	= list(supp = ... , conf = ... , target= ...)。其中 supp 用于设置支持度阈值，默认为 0.1。conf 用于设置信任度阈值，默认为 0.8。target 用于设置返回的规则内容，常设置为 "rules"，以便返回关联规则
appearance	list(rhs = …,default= "lhs") 或 list(lhs = …,default= "rhs") 可通过设置 rhs 或 lhs 来筛选关联规则

下面对前面由疏散矩阵得到的数据 a.class 进行关联分析，并使用 inspect 函数展示详细的关联规则。代码如下：

```
rules<-apriori(a.class,parameter=list(supp=0.2,conf=0.6,target="rules"))
inspect(rules)
    lhs          rhs          support     confidence      lift
1   {}       => {item3}      0.8571429   0.8571429     1.0000000
2   {item2} => {item1}       0.2857143   0.6666667     1.1666667
3   {item2} => {item4}       0.2857143   0.6666667     1.1666667
4   {item2} => {item3}       0.2857143   0.6666667     0.7777778
5   {item1} => {item3}       0.4285714   0.7500000     0.8750000
6   {item4} => {item3}       0.5714286   1.0000000     1.1666667
7   {item3} => {item4}       0.5714286   0.6666667     1.1666667
8   {item2,item4} => {item3} 0.2857143   1.0000000     1.1666667
9   {item2,item3} => {item4} 0.2857143   1.0000000     1.7500000
10 {item1,item4} => {item3} 0.2857143   1.0000000     1.1666667
11 {item1,item3} => {item4} 0.2857143   0.6666667     1.1666667
```

在上述代码中，lhs 列是关联规则的左侧，rhs 列是关联规则的右侧。support、confidence、lift 三列分别是支持度、信任度和提升度。其中关联规则 6 是 9.1.1 节中提到过的。使用 str 函数可以查看 rules 的内部结构，并对其进行分解。代码如下：

```
str(rules)
Formal class 'rules' [package "arules"] with 4 slots
  ..@ lhs    :Formal class 'itemMatrix' [package "arules"] with 3 slots
<中间省略>
..@ quality:'data.frame':     11 obs. of  3 variables:
  .. ..$ support   : num [1:11] 0.857 0.286 0.286 0.286 0.429 ...
  .. ..$ confidence: num [1:11] 0.857 0.667 0.667 0.667 0.75 ...
  .. ..$ lift      : num [1:11] 1 1.167 1.167 0.778 0.875 ...
<后面省略>
```

其中，rules@lhs 返回关联规则左侧的内容，rules@rhs 返回关联规则右侧的内容，rules@quality 返回关联规则的支持度、信任度以及提升度，并使用 rules@quality$support、rules@quality$confidence 以及 rules@quality$lift 分别读取。也可以先把 rules 转化为数据框形式，再读取相应的数据。代码如下：

```
rules.data.frame<- as(rules,"data.frame")
```

如果只关心商品 item3 的推荐规则，则可以只计算 rhs（关联规则的右侧）为 item3 的规则。这时可以把 apriori 函数的 appearance 参数设置为 list(rhs ="item3",default= "lhs")。代码如下：

```
rules<-apriori(a.class,parameter=list(supp=0.2,conf=0.6,target="rules"),appeara
nce=list(rhs ="item3",default= "lhs"))
inspect(rules)
  lhs          rhs          support     confidence   lift
1 {}       => {item3}      0.8571429   0.8571429    1.0000000
```

```
2 {item2} => {item3}       0.2857143  0.6666667  0.7777778
3 {item1} => {item3}       0.4285714  0.7500000  0.8750000
4 {item4} => {item3}       0.5714286  1.0000000  1.1666667
5 {item2,item4} => {item3} 0.2857143  1.0000000  1.1666667
6 {item1,item4} => {item3} 0.2857143  1.0000000  1.1666667
```

然后过滤掉lift小于等于1的关联规则，其中rules@quality返回的是由support、confidence、lift 3列组成的数据框。代码如下：

```
rules<- rules[which(rules@quality$lift>1)]
```

最后使用sort函数并根据某一列排序。例如，下面的代码可以依据lift由大到小让各规则排序，从而得到针对商品item3的最佳推荐（关联）方案。

```
rules.sorted<-sort(rules,by="lift")
```

使用write函数直接把所有关联规则写入txt文件中。代码如下：

```
write(rules,file="apriori_rules.txt",sep = "|",col.names=NA)
```

这里把关联规则写在工作路径下的apriori_rules.txt文件中，并把各个字段使用“|”进行了分隔。

9.2 序列模式关联分析

本节讲解的序列模式关联分析所挖掘的对象以及结果都是有序的，即数据集中每个序列的条目都是有序排列的，且输出的结果也是有序的。例如，如果考虑一个用户多次购物的情况，这些不同时间点的交易记录就构成了一个购买序列，N个用户的购买序列就组成一个规模为N的序列数据集。考虑这些时间上的因素，能得到一些更有价值的规律。例如，一般的关联分析只能挖掘出如啤酒和尿布的推荐销售规律，而序列模式关联分析则能挖掘出诸如用户先购买了《育儿指南》，然后购买婴儿床，之后又购买了婴儿车等这样带有一定因果性质的规律。

9.2.1 序列模式关联分析的原理

1. 引例

表9-3是4名用户的购买记录（包括购买时间在内）。

表9-3 数据集

用户ID	购买时间（日期）	购买商品	用户ID	购买时间	购买商品
1	2013.9.28	A B	2	2013.4.17	A C
	2013.10.8	B		2013.4.27	A B C
	2013.10.18	A B		2013.5.17	B

（续）

用户 ID	购买时间（日期）	购买商品	用户 ID	购买时间	购买商品
3	2012.2.13	A	4	2012.11.6	A B
	2012.3.3	B		2012.11.16	A
	2012.3.13	A		2012.11.26	B

这里不像 9.1 节讲解的一般关联分析那样，以每次购买记录为单位来研究每次购买商品的关联性，而是以每人多次购买记录（考虑购买时间因素）为单位，研究不同顺序下购买商品的关联性。例如，通过粗略观察表 9-3 中的数据可知：用户 1、用户 2、用户 4 均有先购买商品 A 和 B，再继续购买商品 B 的情况，它占所有用户比例的 3/4=75%。因此可以说“先购买商品 A 和 B，再购买商品 B”这样的购买模式很常见，即用户在某次购买了商品 A 和 B 后，接下来很可能会再一次购买商品 B。

2. 数据抽象

为了帮助理解序列模式关联分析的基本原理，下面来介绍几个关键性概念。

序列（sequence）：表示一个完整的信息流，例如，表 9-3 可以表示为表 9-4。

表 9-4 关联模式序列示例

	序列含义	记为
1	用户 1 在不同的时间先后购买了商品 {A,B}，{B}，{A,B}	<{A,B},{B},{A,B}>
2	用户 2 在不同的时间先后购买了商品 {A,C}，{A,B,C}，{B}	<{A,C},{A,B,C},{B}>
3	用户 3 在不同的时间先后购买了商品 {A}，{B}，{A}	<{A},{B},{A}>
4	用户 3 在不同的时间先后购买了商品 {A,B}，{A}，{B}	<{A,B},{A},{B}>

数据项（item）：序列中最小组成单位的集合。例如，表 9-4 中的 {A, B, C} 就是一个数据项。

事件（event）：通常使用时间戳来表示，用于标识同一个序列内不同数据项之间的前后顺序关系。例如，用户 1 的 3 次购买时间依次为 2013.9.28、2013.10.8 和 2013.10.18，两次购买时间间隔每增加 1 天，时间戳增加 1，则 3 次购物时间可以使用时间戳 10、20、30 来表示。最后表 9-3 可以转化为表 9-5。

表 9-5 序列、事件和数据项示例

序列	事件	数据项	序列	事件	数据项
1：<{A,B},{B},{A,B}>	10	{A,B}	3:<{A},{B},{A}>	10	{A}
	20	{B}		30	{B}
	30	{A,B}		40	{A}
2:<{A,C},{A,B,C},{B}>	20	{A,C}	4:<{A,B},{A},{B}>	30	{A,B}
	30	{A,B,C}		40	{A}
	50	{B}		50	{B}

3. 序列的包含关系、支持度及序列模式关联分析

理解了序列、数据项和事件的概念后，再介绍序列的包含关系以及支持度的概念。

序列的包含关系：对于序列 x 和 y，如果存在一个映射关系，使得 x 中的每个事件都包含于 y 中，则称为 x 被包含于 y（x 是 y 的子序列）。例如，序列 <{B},{A,C}> 是序列 <{A,B},{E},{A,C,D}> 的子序列。

支持度（support）：某序列 x 的支持度是指在整个序列集合中包含 x 的序列频次。例如，由于每个序列中均包含序列 <{A}>，所以序列 <{A}> 的支持度为 4/4=1；而序列 <{A,B},{B}> 分别被 <{A,B},{B},{A,B}>、<{A,C},{A,B,C},{B}> 以及 <{A,B},{A},{B}> 这 3 个序列所包含，所以其支持度为 3/4=0.75。

序列模式关联分析：假设 min_support 是最小支持度阈值，则序列模式关联分析就是找出所有支持度大于 min_support 的序列。常用的算法为由 zaki 提出的可以使用各种约束限制的 cSPADE 算法。

9.2.2 序列模式关联分析的 R 语言实现

1. cspade 函数

R 语言实现 cSPADE 算法的核心函数是 arulesSequences 包的 cspade 函数。其中，参数 parameter 用于设置各类约束限制。cspade 函数的参数说明如表 9-6 所示。

表 9-6 cspade 函数的参数说明

参数	说明
support	支持度阈值（最小值），取值范围是 [0,1]，默认值为 0.1
maxlen	返回的结果序列中包含数据项的最大数，默认值为 10 例如，如果设置 maxlen =2，则序列 <{A,B,C},{B}> 符合条件，序列 <{A,B,C},{B},{A}> 不符合条件（序列中包含了 {A,B,C},{B},{A}3 个数据项）
maxsize	返回结果序列的数据项中所包含商品的最大数，默认值为 10 例如，如果设置 maxsize =2，则序列 <{A,B},{B}> 符合条件，序列 <{A,B,C},{B}> 不符合条件（序列中的 {A,B,C} 包含了 3 个商品）
maxgap	同一序列内前后事件（event）间隔的最大值 例如，不设置 maxgap 参数时，序列 1、序列 2、序列 4 包含序列 <{A,B},{B}>，即支持度为 3/4=0.75。设置 maxgap=15 时，第 2 个序列 <{A,C},{A,B,C},{B}> 中的 event 依次为 20、30、50。因此在 maxgap=15 时，不再考虑第 2 个序列的子序列 <{A,C},{B}> 和 <{A,B,C},{B}>，而只考虑 <{A,C},{A,B,C}> 这个子序列的情况，最后导致在 maxgap=15 时，序列 2 不再包含序列 <{A,B},{B}>。同理第 4 个序列 <{AB},{A},{B}> 也是只考虑 <{AB},{A}> 和 <{A},{B}> 的情况，并且均不包含 <{A,B},{B}>，因此它也不符合条件。而第 1 个序列 <{A,B},{B},{A,B}> 可以考虑 <{A,B},{B}> 和 <{B},{A,B}> 子序列的情况，并且 <{A,B},{B}> 包含了目标子序列 <{A,B},{B}>，因此在 maxgap=15 时，序列 <{A,B},{B}> 的支持度 =1/4=0.25
mingap	同一序列内前后事件（event）间隔的最小值，具体可以参照 maxgap 的参数说明
maxwin	同一序列内任意事件（event）间隔的最大值 例如，不设置 maxwin 参数时，序列 1、序列 2、序列 4 包含序列 <{A,B},{B}>，即支持度为 3/4=0.75。设置 maxwin=15 时，在第 2 个序列 <{A,C},{A,B,C},{B}> 中，{A,B,C} 和 {B} 对应 event 的差为 50-30=20>15，因此在 maxwin=15 时，序列 <{A,C},{A,B,C},{B}> 不再包含序列 <{A,B},{B}>。同理第 4 个序列 <{AB},{A},{B}> 也是如此。而只有第 1 个序列 <{A,B},{B},{A,B}> 中 {A,B} 和 {B} 对应 event 的差为 20-10=10<=15 符合条件（注意 {A,B} 和 {A，B} 对应 event 的差为 30-10=20>15 也不符合条件）。因此在 maxwin=15 时，序列 <{A,B},{B}> 的支持度 =1/4=0.25

2. 引例实现

下面使用 cspade 函数对表 9-3 中提到的数据进行分析。

（1）数据准备

1）把表 9-3 中的原始数据转化为图 9-3 中表 A 的形式（注意，这里事件 event 的设置没有固定的形式，只要可以表示同一序列内不同数据项间的先后顺序便可），并存入 R 语言对象 data.train 中，如图 9-3 所示。

表A

数据项 (item)	序列 (sequence)	事件 (event)
A	1	10
B	1	10
B	1	20
A	1	30
B	1	30
A	2	20
C	2	20
A	2	30
B	2	30
C	2	30
B	2	50
A	3	10
B	3	30
A	3	40
A	4	30
B	4	30
A	4	40
B	4	50

代码以及输出展示

```
item<-factor(c("A","B","B","A","B","A","C","A","B","C","B","A","B",
        "A","A","B","A","B"))
seqid<- c(1,1,1,1,1,2,2,2,2,2,2,3,3,3,4,4,4,4)
eventid<-c(10,10,20,30,30,20,20,30,30,30,50,10,30,40,30,30,40,50)
data<-data.frame(item=item)
data.tran<-as(data,"transactions")
transactionInfo(data.tran)$sequenceID<- seqid
transactionInfo(data.tran)$eventID<- eventid
transactionInfo(data.tran)
```

	transactionID	sequenceID	eventID
1	1	1	10
2	2	1	10
3	3	1	20
4	4	1	30
5	5	1	30
6	6	2	20
7	7	2	20
8	8	2	30
9	9	2	30
10	10	2	30
11	11	2	50
12	12	3	10
13	13	3	30
14	14	3	40
15	15	4	30
16	16	4	30
17	17	4	40
18	18	4	50

图 9-3 数据集和代码示例

2）把表 A 中的数据存于向量中，注意商品列表（item 信息向量）必须为分类变量。代码如下：

```
    item<-factor(c("A","B","B","A","B","A","C","A","B","C","B","A","B","A","A","B",
"A","B"))
    seqid<- c(1,1,1,1,1,2,2,2,2,2,2,3,3,3,4,4,4,4)
    eventid<-c(10,10,20,30,30,20,20,30,30,30,50,10,30,40,30,30,40,50)
```

3）使用 item 向量建立一个数据框 data，并使用 as(data,"transactions") 函数将其转化为 cspade 函数可使用的数据对象 data.tran。代码如下：

```
    data<-data.frame(item=item)
    data.tran<-as(data,"transactions")
```

4）依次添加好序列编号和事件数值。代码如下：

```
transactionInfo(data.tran)$sequenceID<- seqid
transactionInfo(data.tran)$eventID<- eventid
```

并使用 transactionInfo(data.tran) 查看最后结果。

（2）cspade 函数处理

下面使用 cspade 函数进行序列模式关联分析处理，这里参数 parameter 的设置意义已经在 9.2.2 节中讲解过，另外，control 参数通常设置为 list(verbose = TRUE)。可通过 sort 函数对 support 支持度进行由大到小的排序，并使用 inspect 函数查看结果内容。代码如下：

```
result<-cspade(data.tran,parameter = list(support = 0.5),control = list(verbose
= TRUE))
result<-sort(result,by="support")
inspect(result)
```

如果很关心商品 B 的关联情况，则可以使用 grep 函数并结合规则表达式来筛选出与商品 B 相关的关联序列。其中，grep(a,b) 函数可以返回向量 / 列表 b 中包含字符串 a 的序列号。由于需要通过 result 的 sequence 列来筛选，所以 b 等于 as(result,"data.frame")$sequence)。而字符串 a 则需要根据不同的研究目的来确定（需要使用规则表达式）。最后可使用 result[grep(a,b)] 筛选出包含字符串 a 的序列，如表 9-7 所示。

表 9-7 规则表达式的设置

	研究目的	设置 a 的规则表达式
1	研究所有购买了商品 B 的序列	".*item=B.*"
	. 表示除空格符外的任何字符串，* 表示重复 0 到多次。最后可以匹配出包含 item=B 的序列，即用户购买了商品 B 的序列，代码如下： result.1<-result[grep(".*item=B.*",as(result,"data.frame")$sequence)]	
2	哪些购买行为会最后促使用户购买商品 B	".*item=B[^\\}]*\\}>"
	[^\\}] 表示匹配任何不匹配 } 的字符，\\} 表示匹配 } 的字符，最后可以筛选出最后一个数据项内包含 item=B(最后一个 {} 中包含 item=B) 的序列。代码如下： result.2<-result[grep(".*item=B[^\\}]*\\}>",as(result,"data.frame")$sequence)]	
3	用户购买了商品 B 后，后续还会购买哪些商品	"<\\{[^\\{]*item=B.*"
	\\{ 表示匹配 { 的字符，[^\\{] 表示匹配任何不匹配 { 的字符，最后可以筛选出第一个数据项内包含 item=B(第一个 {} 中包含 item=B) 的序列。代码如下： result.3<-result[grep("<\\{[^\\{]*item=B.*",as(result,"data.frame")$sequence)]	

9.2.3 实例：用户点击页面的行为分析

在网站分析中，可以使用序列模式的关联分析来研究用户的点击行为流，并衡量页面的体验以及易用性效果。这里使用 www.microsoft.com 网站的用户浏览页面轨迹数据研究：哪些页面是引导用户进入关键页面 1034 的重点页面，以及其引导能力等问题。

1. 数据集介绍

数据集 anonymous-msweb 记录了 1998 年 2 月某周内用户访问 www.microsoft.com 页面的行为数据，其下载链接为 http://kdd.ics.uci.edu/databases/msweb/msweb.html。这里使用训

练集数据 anonymous-msweb.train 文件，用 vim 编辑器打开，其格式如图 9-4 所示。

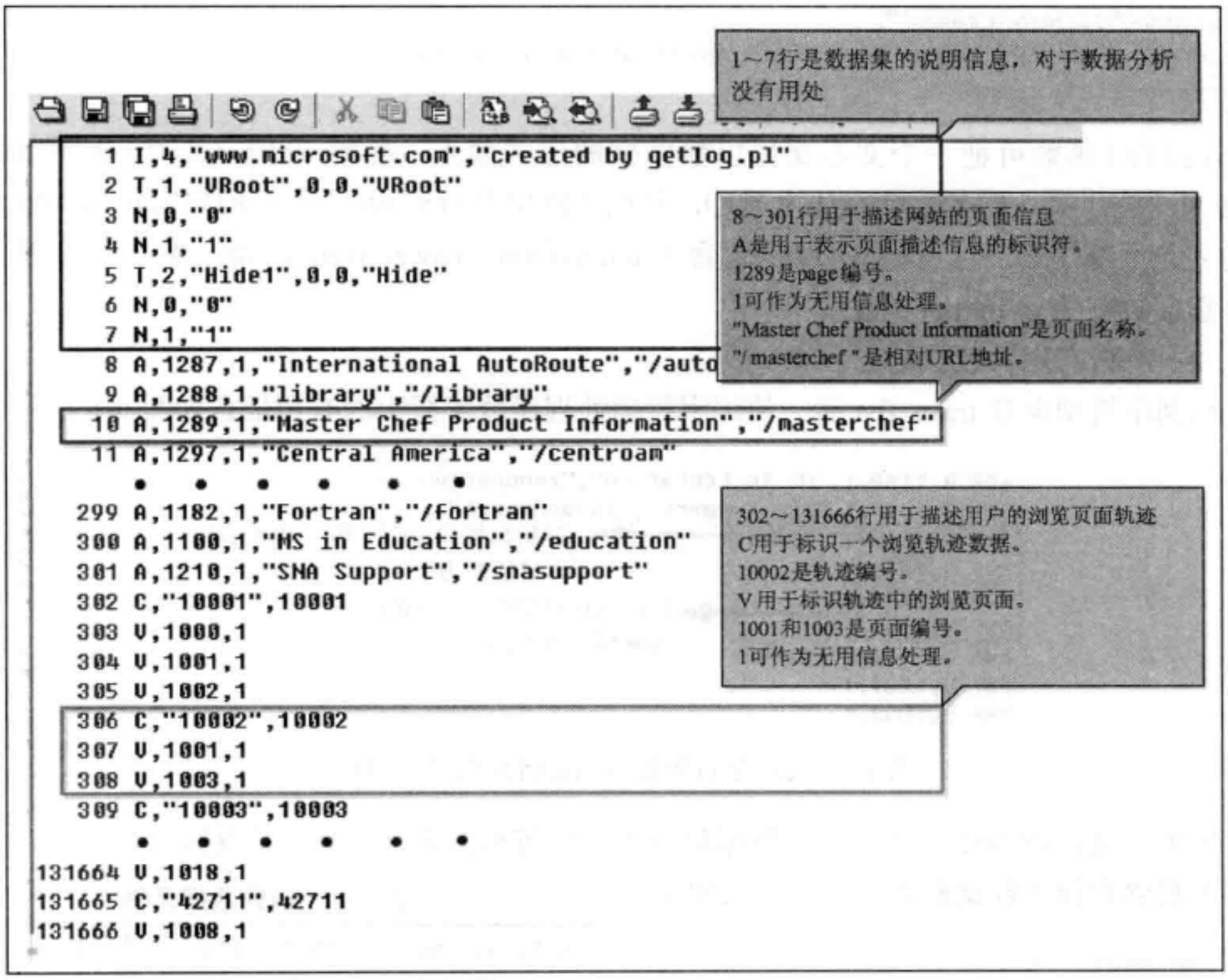


图 9-4 数据集说明

其中，1 ～ 7 行是文件的第一部分，是数据集的说明信息，对于数据分析没有用处。

8 ～ 301 行用于描述网站的页面信息，每行可描述一个页面信息，格式为：A 是用于表示页面描述信息的标识符，然后是页面的编号，1 可作为无用信息处理，之后是页面名称，最后是相对 URL 地址。

302 ～ 131666 行用于描述用户的浏览页面轨迹。以 C 开头，用以标识一个浏览轨迹数据，随后是轨迹编号。接下来的几行用以描述本次轨迹中浏览过的页面信息。V 用以标识轨迹中的浏览页面，然后是页面编号，最后的 1 可作为无用信息处理。

2. 读取数据

了解了数据集文件 anonymous-msweb.train 的格式和意义后，下面把相关信息读入 R 语言环境中。

（1）使用 read.fwf 函数读入文件

数据文件的本地路径为 "C:\web_trace\data\train\anonymous-msweb.train"（Windows 环境下），这里把 "C:\web_trace\" 设置为工作目录。注意，在 R 语言中，路径的分隔符为“ /”，

与 Windows 环境下的“\”不同。

```
root<-"C:/web_trace/"
tmpp <-read.fwf(paste(root,"data/train/anonymous-msweb.train",sep=""),widths= c(60))
train_list<-tmpp$V1
```

read.fwf 函数可把一个文本文件以数据框的形式读入，参数 widths=c(60)) 表示数据框 tmpp 只含有 1 列（列名称是默认为 V1），并把文件中每行前 60 个字符的内容作为 tmpp$V1 中的一个元素。最后 tmpp$V1 就是包含了 anonymous-msweb.train 内每行数据的字符型向量，它会被赋予 train_list 向量。

（2）解析有用的数据信息

得到字符型向量 train_list 后，使用字符串处理函数解析出的有用信息如图 9-5 所示。



图 9-5 使用字符串处理函数解析出的有用信息

例如，文件内 302 ～ 308 行的数据最后可分别解析为表 9-8 所示的数据。并最终存储于数据框 data 中。代码如下：

表 9-8 部分数据示例

序列 ID：tmp_sequenceid	事件 ID：tmp_eventid	页面编号（数据项）：tmp_page
1	1	1000
1	2	1001
1	3	1002
2	1	1001
2	2	1003

```
tmp_page<-c(0)
tmp_sequenceid<-c(0)
tmp_eventid<-c(0)
m<-0
sequenceid<-0
train_length<-length(train_list)
for (i in 302:train_length) {
    tmp<-unlist(strsplit(as.
    character(train_list[i]), ","))
    if(tmp[1]=="C") {
        sequenceid<-sequenceid+1
        eventid<-0
    } else if(tmp[1]=="V") {
        m<-m+1
        eventid<-eventid+1
        tmp_sequenceid[m]<-sequenceid# 填充 sequenceID
        tmp_eventid[m]<-eventid# 填充 eventID
        tmp_page[m]<-as.numeric(tmp[2])
    }
}
tmp_page<-factor(tmp_page)
data<-data.frame(page=tmp_page,seqid=tmp_sequenceid,eventid=tmp_eventid)
```

上述代码分别建立向量 tmp_sequenceid、tmp_eventid 以及 tmp_page，以便分别存储序列、事件、数据项等信息。变量 m 是用于向量序号的计数标识，在读取到 V 标示符后自动计数加 1。i 是主循环程序的计数器，用于读取文件中的所有用户轨迹行为数据（第 302 行至最后一行间的数据），每读取完一行信息，i 自动计数加 1。以下代码是解析信息的核心部分。

```
tmp<-unlist(strsplit(as.character(train_list[i]), ","))
```

上述代码使用 strsplit 函数分隔每行字符串信息 as.character(train_list[i])，逗号“,”为间隔符，从而得到一个列表对象。然后使用 unlist 函数把该列表对象变为向量数据，并赋值予 tmp 对象内。例如，i=302 和 303 时（即解析文件中 302 和 303 行的数据时），向量 tmp 的内容如表 9-9 所示。

表 9-9 数据解析

	i=302	i=303
train_list[i]	字符串 C,"10001",10001	字符串 V,1000,1
tmp	字符型向量："C" "\"10001\"" "10001"	字符型向量："V" "1000" "1"

解析出的向量 tmp 的含义是：tmp[1] 是标示符，如果为 C，则表示一条新轨迹的开始。如果是 V，则表示一个页面信息，此时 tmp[2] 表示页面编号。如果读取的标示符是新轨迹标示符 C，则序列号 ID 加 1，事件 ID 清零。代码如下：

```
if(tmp[1]=="C") {
    sequenceid<-sequenceid+1
    eventid<-0
```

如果读取的标示符是页面信息标示符 "V"，则事件 ID 自动加 1，并把 tmp[2] 赋值给数据项 tmp_page。代码如下：

```
} else if(tmp[1]=="V") {
    m<-m+1
    eventid<-eventid+1
    tmp_sequenceid[m]<-sequenceid# 填充 sequenceID
    tmp_eventid[m]<-eventid# 填充 eventID
    tmp_page[m]<-as.numeric(tmp[2])
}
```

由于关联分析时数据项必须是分类变量，所以使用 factor 函数把向量 tmp_page 转化为分类变量。最后把向量 tmp_sequenceid、tmp_eventid 和 tmp_page 统一存储于数据框 data 中。代码如下：

```
tmp_page<-factor(tmp_page)
data<-data.frame(page=tmp_page,seqid=tmp_sequenceid,eventid=tmp_eventid)
```

（3）筛选浏览轨迹中包括目标页面 1034 的数据

因为本实例要寻找哪些页面会引导用户进入关键页面 1034，以及其贡献度和能力。所

以这里先要筛选出轨迹中与目标页面 1034 相关的数据 data.user。代码如下：

```
user.page=1034
user.sequenceid<-unique(data$seqid[which(data$page==user.page)])
i<-1;data.user<-data[1,]
for(seq.i in user.sequenceid){
    data.user<-rbind(data.user,subset(data,seqid==seq.i))
    i<-i+1
}
data.user<-data.user[-1,]
```

user.page 用于设置目标页面编号为 1034。which(data$page==user.page) 用于找出 page 等于 1034 的行数据序号，并通过 unique(data$seqid[..]) 得到经去重处理后的所有包含页面 1034 的序列编号 user.sequenceid。然后依次轮询 user.sequenceid 内的序列号 seq.i，使用 subset(data,seqid==seq.i) 筛选出 data 的 seqid 列等于 seq.i 的数据，并使用 rbind 函数逐次扩展于数据框 data.user 中。

3. 使用 cspade 函数进行序列模式关联分析

（1）数据准备

首先加载 arulesSequences 算法包，并把存有目标页面 1034 浏览轨迹信息（包括序列 id 和事件 id 以及数据项信息）的数据框 data.user 转化为 cspade 函数可用的数据对象 data.tran。代码如下：

```
library(arulesSequences)
tmp_data<-data.frame(page=data.user$page)
data.tran<-as(tmp_data,"transactions")
transactionInfo(data.tran)$sequenceID<-data.user$seqid
transactionInfo(data.tran)$eventID<-data.user$eventid
```

（2）使用 cspade 函数进行分析

找出用户进入 1034 页面前浏览过的页面 i，以及序列 <{page=i},{page=1034}> 的支持度。代码如下：

```
    result<-cspade(data.tran,parameter=list(support=0,maxlen=2),control=list(verbos
e=TRUE))
    result<-sort(result,by="support")
    page.2<-paste(".*page=",user.page,"[^\\}]*\\}>",sep="")
    result.2<-result[grep(page.2,as(result,"data.frame")$sequence)]
```

由于要找出所有页面，所以将支持度阈值 support 设置为 0。同时由于欲返回序列的数据项数最大为 2，所以 maxlen 被设置为 2，即 cspade 函数的 parameter=list(support=0,maxlen=2)。然后使用 sort 函数并依据支持度 support 的数值将 result 进行排序，并设置规则表达式 paste(".*page=",user.page,"[^\\}]*\\}>",sep="")，即 ".*page=1034[^\\}]*\\}>"，筛选出序列中最后一个数据项为 {page=1034} 的序列。最后得到的 result.2 格式如下：

```
inspect(result.2)
       items                 support
  1 <{page=1034}>      1.00000000
  2 <{page=1008},
     {page=1034}>      0.55472663
  3 <{page=1017},
     {page=1034}>      0.16082276
                < 中间省略 >
240 <{page=1284},
    {page=1034}>      0.0001065757
```

其中序列 1 为 <{page=1034}>，无用，之后的 239 个序列是我们希望得到的序列 <{page=i},{page=1034}> 的信息。

注意

序列 <{page=i},{page=1034}> 的支持度 support 可用于衡量页面 i 对关键页面 1034 浏览次数 uv 的贡献度。

4. 筛选重点页面并衡量其引导能力

（1）重点页面

1）计算各页面支持度的百分比，并把比例累计达到 70% 的页面作为引导用户进入 1034 页面的重点页面。为了方便读取 result.2 内的数据，这里把 result.2 转化为数据框 result.data.frame。代码如下：

```
result.data.frame<-as(result.2[-1],"data.frame")
```

其中，result.2[-1] 用于剔除无用序列 <{page=1034}> 的信息。

2）计算支持度 support 的百分比。代码如下：

```
persent<-result.data.frame$support/sum(result.data.frame$support)
```

3）使用累加函数 cumsum() 计算支持度 support 的累计百分比，并把百分比数据 persent 以及 sum.persent 并入数据框 result.data.frame 中。代码如下：

```
sum.persent<-cumsum(persent)
result.data.frame<-cbind(result.data.frame,persent,sum.persent)
```

4）使用 subset 函数筛选出累计百分比 sum.persent 小于 70%(max.persent) 的序列数据。代码如下：

```
max.persent=0.7
result.data.frame<-subset(result.data.frame,sum.persent<=max.persent)
```

最终数据框 result.data.frame 的格式如下：

```
result.data.frame
                          sequence    support      persent      sum.persent
6458 <{page=1008},{page=1034}>    0.55472663    0.24289514    0.2428951
6467 <{page=1017},{page=1034}>    0.16082276    0.07041859    0.3133137
                        <中间省略>
6516 <{page=1070},{page=1034}>    0.02749654    0.01203976    0.6881796
6479 <{page=1030},{page=1034}>    0.02568475    0.01124644    0.6994260
```

它包含了重点页面的序列信息（sequence）、支持度（support），以及支持度的（累计）百分比（persent 和 sum.persent）。

（2）解析重点页面编号

列数据 result.data.frame$sequence 是 <{page=i},{page=1034}> 形式的序列信息，下面解析出其中的页面编号 i，并存放于 page 向量中。代码如下：

```
page<-0;i<-1
for(i.seq in result.data.frame$sequence){
    real_seq1<-regexpr("<\\{page=",i.seq)+7
    real_seq2<-regexpr("\\}",i.seq)[1]-1
    page[i]<-substr(i.seq,real_seq1,real_seq2)
    i<-i+1
}
```

其中，real_seq1 是字符串 <{page=i},{page=1034}> 中页面编号 i 的开始位置信息，real_seq2 是结尾位置信息，最后使用 substr 函数直接截取出此段字符串内容，并存储于向量 page 中。

（3）重点页面的引导能力

以上只研究了哪些重点页面对关键页面 1034 浏览次数的贡献度较大。然而如果页面 i 的浏览次数本身就很大，则其作为重点页面的概率也会较大。这里使用：

conf(i=>1034)= 页面 i 引导用户进入关键页面 1034 的次数 / 页面 i 的浏览次数来衡量页面 i 引导用户进入页面 1034 的能力，由于其含义类似于信任度，因此这里记为 conf。得到重点页面编号 page 后，可以通过原始数据 data 来计算页面 i 的浏览次数。代码如下：

```
i<-1;uv<-0
for(i.page in page){
    uv[i]<-length(unique(data$seqid[which(data$page==i.page)]))
    i<-i+1
}
```

页面 i 引导用户进入关键页面 1034 的次数就是序列 <{page=i},{page=1034}> 的支持度乘以其序列数 result@info$nsequences，最后得到：

```
conf<-result.data.frame$support* result@info$nsequences/uv
```

把 page 和 conf 也并入数据框 result.data.frame 中。代码如下：

```
result.data.frame<-cbind(result.data.frame,conf=conf,page=page)
```

注意

conf(i=>1034)= 页面 i 引导用户进入关键页面 1034 的次数 / 页面 i 的浏览次数，用于衡量页面 i 引导用户进入页面 1034 的能力。

5. 绘制结果图表

首先使用柱状图 barplot 函数绘制序列 <{page= 重要页面 i},{page=1034}> 支持度的百分比 result.data.frame$persent，然后使用红色曲线（lines 函数）绘制衡量重要页面 i 引导用户能力的 conf，接着使用 text 函数在曲线上绘制 conf 的数值，最后使用 axis 函数把重要页面的编号信息填 x 轴处。代码如下：

```
barplot(as.matrix(result.data.frame$persent,nrow=1),ylim=c(0,1),beside=TRUE,xlab=
"页面名称",main="引导用户进入关键页面 1034 的重点页面分析")
lines(0.5+c(1:nrow(result.data.frame)),result.data.frame$conf,type="b",col="red")
text(0.5+c(1:nrow(result.data.frame)),result.data.frame$conf,labels=paste(round
(result.data.frame$conf*100,2),"%",sep=""))
axis(1,at=0.5+c(1:nrow(result.data.frame)),labels=result.data.
frame$page,tick=FALSE)
```

绘制结果如图 9-6 所示。

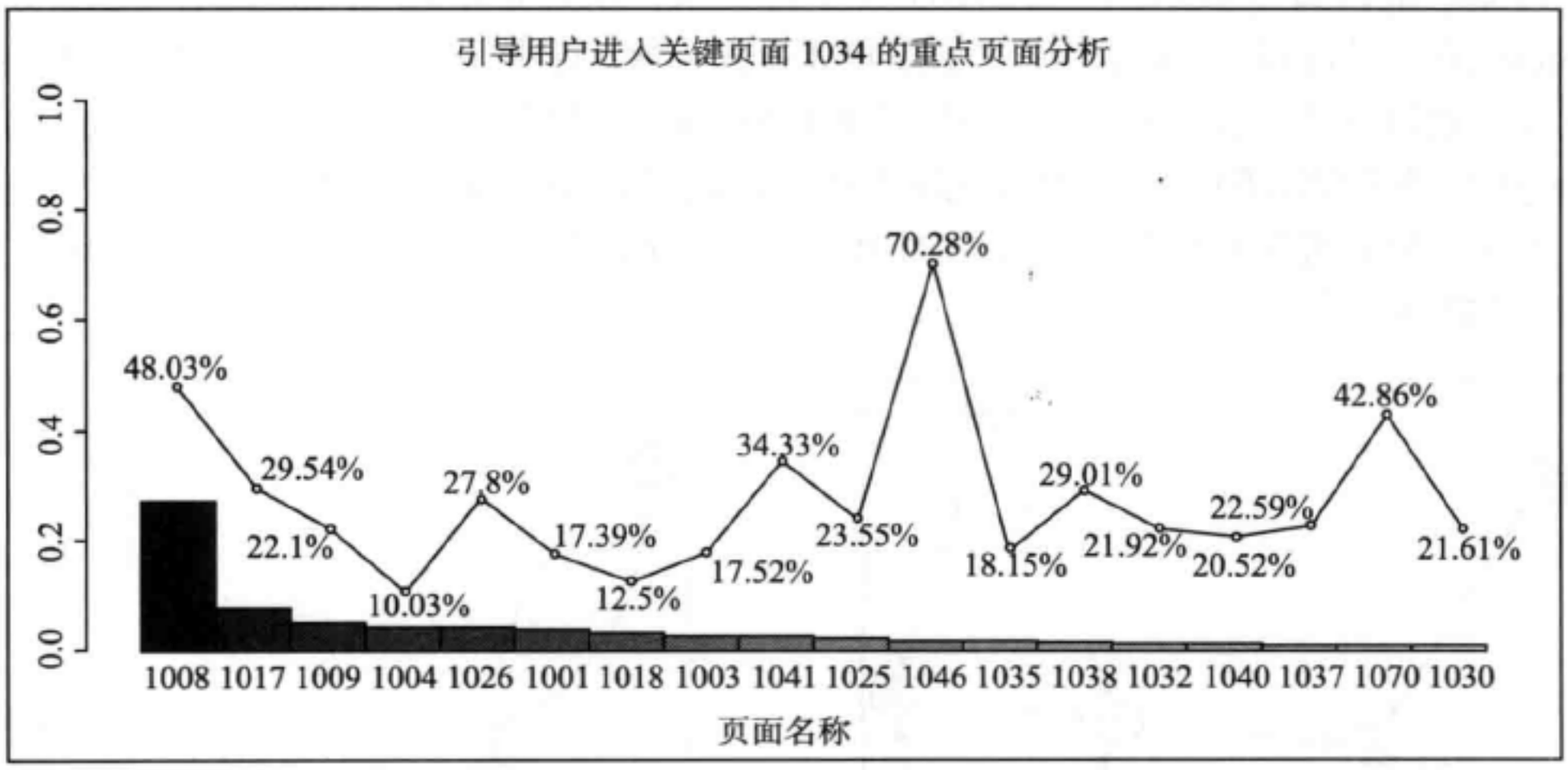


图 9-6 重点引导页面分析图

结论：页面 1008 是为页面 1034 的访问贡献最大的页面，其次是 1017、1009 等。而在引导能力上，页面 1004 和 1018 较低，鉴于页面 1004 为关键页面 1034 的访问贡献较大（排名第 4），但引导能力较低，因此要重点优化该页面。

9.3 关系网络分析

随着以社交和资讯分享为主的社会化网站的兴起（如 facebook、人人网和新浪微博等），社会网络分析（Social Network Analysis，SNA）逐步成为数据挖掘领域的又一新宠。SNA 的本质是利用各样本间的关系（故也称为关系网络，以后相关章节将以此命名）来分析整体样本的群落现象，并分析出样本点在群落形成中的作用以及群落间的关系。本节将介绍社会网络分析的基本概念和原理，并讲解如何使用 R 语言的 igraph 包实现 SNA。

9.3.1 网络关系分析的基本概念

1. 引例

表 9-10 是某社交网站上 7 位用户 1 天里的留言记录，每一个留言 id 表示一次留言行为。

表 9-10 数据集

留言 id	发布留言用户	接收留言用户	留言 id	发布留言用户	接收留言用户
1	a	c	6	c	d
2	a	e	7	d	g
3	e	c	8	d	g
4	b	e	9	d	f
5	b	c	10	f	d

例如，留言 id=9 表示用户 d 给用户 f 留言一次，用户 d 和 f 有一个关系，把该关系记为 edge(d,f)。留言 id=10 表示用户 f 给用户 d 留言一次（可能是回复留言），用户 d 和 f 有一个关系，把该关系记为 edge(f,d)。如果想研究用户间主动留言行为（即区分关系 edge(d,f) 和 edge(f,d)），则要使用有向图（图中使用箭头来标识留言行为的发起方和接收方），如图 9-7 所示。

如果只想研究用户间是否发生过沟通（即不区分关系 edge(d,f) 和 edge(f,d)），则使用无向图，如图 9-8 所示。

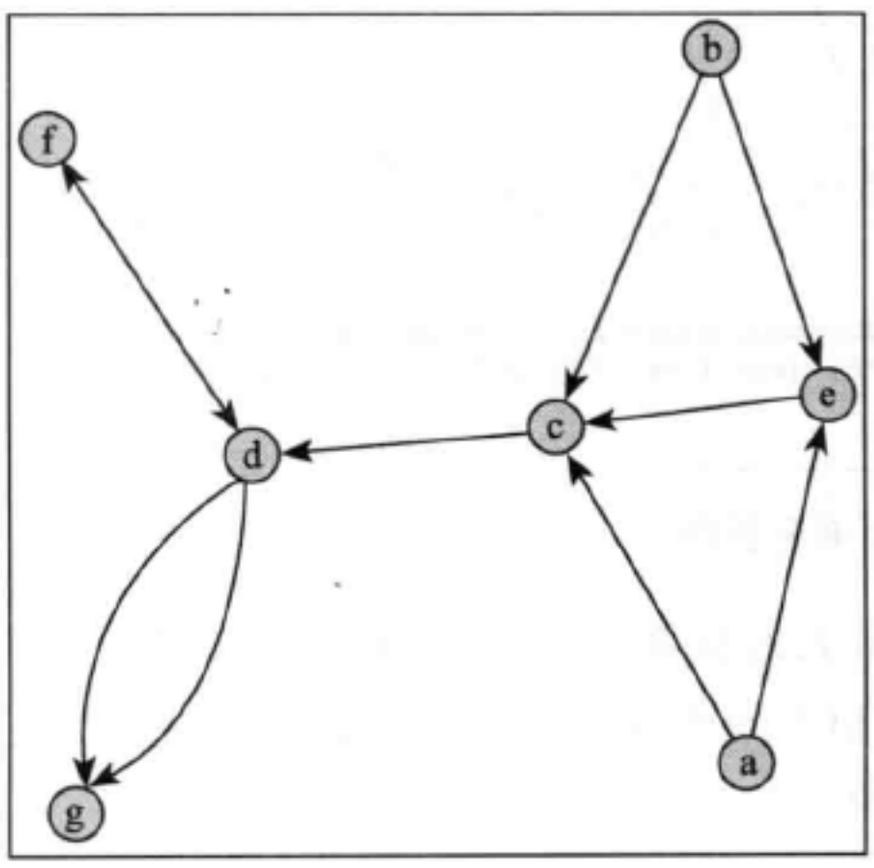


图 9-7 有向图示例

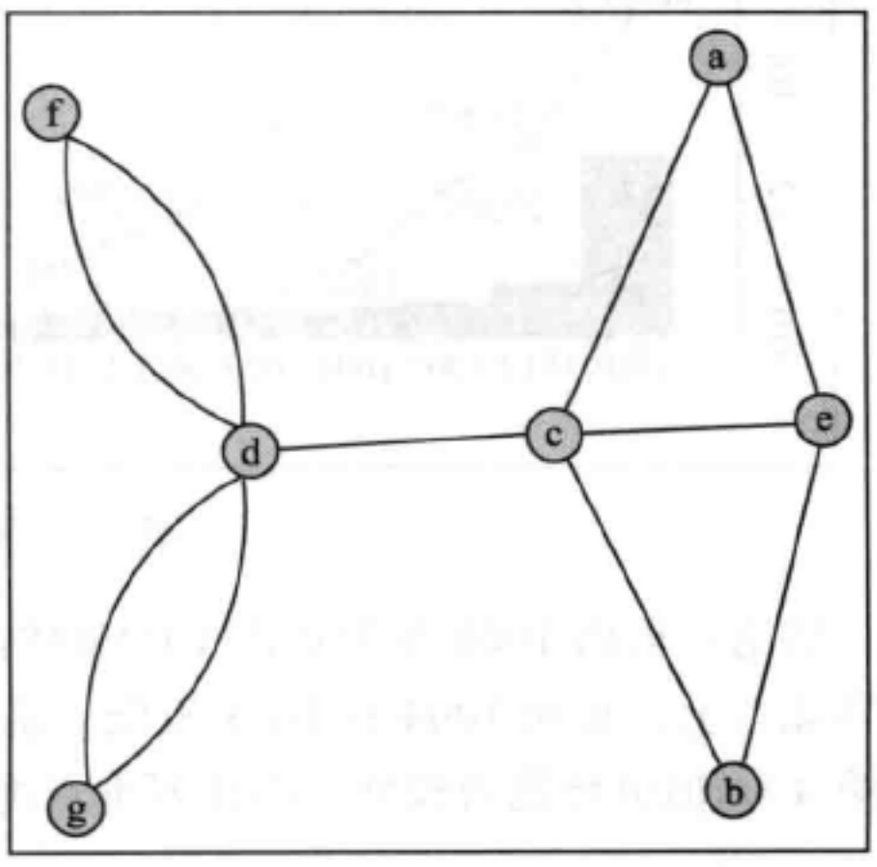


图 9-8 无向图示例

关系网络图无非是一些点和线的组合，最后可以抽象出如表 9-11 所示的网络关系分析基础概念。

表 9-11 网络关系分析基础概念

概念	说明
V	关系图中，所有点（vertices）的集合，图 9-8 中点的集合是 V={a,b,c,d,e,f,g}
v_i	一个点，即某一样本
n	图中点的数目 =\| V \|
N_i	与点 v_i 相邻的点的集合。 有向图：相邻入点为直接指向该点的点集合，如点 d 相邻入点 N_d={f,c}。 相邻出点为该点直接指向的点集合，如点 d 相邻出点 N_d={f,g}。 无向图：与该点相连的点集合，如点 d 相邻的点 N_d={f,c,g}
d_i	点 v_i 的度数（degree），等于与点 v_i 相邻的点的总数 =\| N_i \|。 有向图：入度为直接指向该点的点总数，如点 d 的入度 d_d=2。 出度为该点直接指向的点总数，如点 d 的出度 d_d=2。 无向图：与该点相连的点的总数，如点 d 相邻的点 d_d=3
E	关系图中所有线（edges）的集合，用于标识样本间产生过的关系
edges(v_i,v_j)	连接点 v_i 和 v_j 之间的连线。在有向图中表示该关系是由点 v_i 发起，由点 v_j 接收（射线 $v_i \rightarrow v_j$）。在无向图中表示点 v_i 和 v_j 产生了关系（直线 vi----vj）
weigth(v_i,v_j)	线的权重，等于两点间线的数目。在图 9-8 中，weigth(d,f)=2 ；weigth(d,g)=2。在图 9-7 中，weigth(d,g)=2（注意 weigth(d,f)=1； weigth(f,d)=1 ）
m	图中线的数目 =\| E \|
shortest(vi,vj)	连接点 v_i 和 v_j 之间的最短路径。例如，在无向图中，点 d 和 e 之间的最短路径为 shortest(d,e)=d → c → e。最短路径也称为测地线
g(v_i,v_j)	点 v_i 和 v_j 之间的距离（测地线距离），等于 shortest(v_i,v_j) 中包含线的数量。例如，在无向图中，点 d 和 e 之间的距离 g(d,e)=2。而点 c 和 f 之间的最短路径是 c → d → f，c 和 d 间包括 1 条线，d 和 f 间包括 2 条线（线权重为 2），所以距离 g(c,f)=3

通过线 edges(v_i,v_j) 的概念可以了解两个不同样本点是否产生了直接关系，而线的权重 weigth(v_i,v_j) 可以衡量两个不同样本点的直接关系大小。通过点之间的距离 g(v_i,v_j) 可以衡量两个不同样本点的间接关系大小。某一点的度（degree）可以衡量该点与整个网络的联系程度。

2. 用 R 语言建立关系网络

使用 igraph 算法包来建立 9.3.1 节引例中的关系网络，并计算表 9-11 中的相关概念。

（1）原始数据准备

把表 9-10 中的数据读入 R 环境中。代码如下：

```
from<-c("a","a","e","b","b","c","d","d","d","f")
to<-c("c","e","c","e","c","d","g","g","f","d")
data<-data.frame(from=from,to=to)
```

这里把每条线 edges(v_i,v_j) 的 v_i 组合为 from 向量，v_j 组合为 to 向量，并存放于 data 数据框中。然后读取每个样本点的名称：

```
labels<-union(unique(data[,1]),unique(data[,2]));labels
[1]  "a"    "e"    "b"    "c"    "d"    "f"    "g"
```

这里分别把 from(data[,1]) 和 to(data[,2]) 向量的元素去重，再通过集合关系中的合并函数进行合并，得到关系网络中所有样本点的名词集合 labels。建立点标识号向量 ids，并把其代表的样本名称 labels 作为元素名称。代码如下：

```
ids<-1:length(labels)
names(ids)<-labels;ids
a   e   b   c   d   f   g
1   2   3   4   5   6   7
```

通过向量 ids 中元素值和元素名称的映射关系，把数据框 data 转化为矩阵 edges。代码如下：

```
from<-as.character(data[,1]);to<-as.character(data[,2])
edges<-matrix(c(ids[from],ids[to]),nc=2);
```

最后实现的结果如图 9-9 所示。

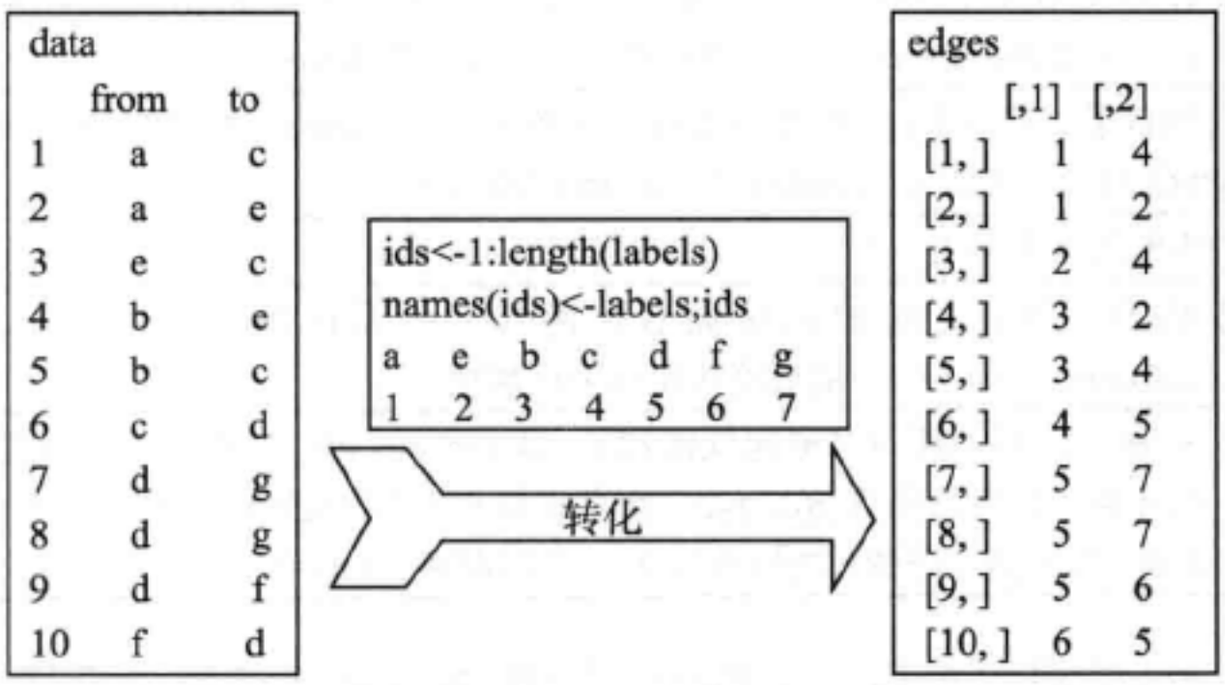


图 9-9 转化示例

（2）建立关系网络

1）使用 graph.empty 函数建立一个空的关系网络 g（没有定义任何点和线的信息），当参数 directed=TRUE 时，该关系网络为有向图，当 directed=FALSE 时为无向图。这里以无向图为例。

```
g<-graph.empty(directed=F)
```

2）使用 add.vertices 函数向关系网络 g 中添加 length(labels) 个点 vertices，并为各个点添加名称信息 V(g)$label。代码如下：

```
g<-add.vertices(g,length(labels))
V(g)$label=labels
```

3）使用 add.edges 函数向关系网络 g 中添加线 edges 信息。代码如下：

```
g<-add.edges(g,t(edges))
```

注意，这里的 edges 要先转置为如下形式：

```
t(edges)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    1    2    3    3    4    5    5    5     6
[2,]    4    2    4    2    4    5    7    7    6     5
```

才能作为 add.edges 函数的输入数据。

4）把重复的线合并，并把重复次数作为线权重 weigth。代码如下：

```
E(g)$weight <- count.multiple(g);E(g)$weight
  [1] 1 1 1 1 1 1 2 2 2 2
    g<-simplify(g, remove.multiple = TRUE, remove.loops = TRUE,edge.attr.comb =
'mean')
    E(g)$weight
    [1] 1 1 1 1 1 1 2 2
```

这里的 count.multiple(g) 用于计算网络 g 中每条线的重复数。对于无向图 g，线 7 和线 8 之间是重复的（均为 edge(d,g)），所以其 count.multiple 值为 2。同理，线 9 和线 10 也是如此（均为 edge(d,f)）。使用 simplify 函数把图中重复的线删除（remove.multiple=TRUE），并使用重复的次数来更新线权重 E(g)$weight(edge.attr.comb = 'mean')。

至此，就建立了一个完整的关系网络 g，下面提取上述代码中的部分对象作为参数，并给出一段自定义的函数 init.igraph 用于建立关系网络。代码如下：

```
    init.igraph<-function(data,dir=F,rem.multi=T){
        labels<-union(unique(data[,1]),unique(data[,2]))
        ids<-1:length(labels);names(ids)<-labels
        from<-as.character(data[,1]);to<-as.character(data[,2])
        edges<-matrix(c(ids[from],ids[to]),nc=2)
        g<-graph.empty(directed=dir)
        g<-add.vertices(g,length(labels))
        V(g)$label=labels
        g<-add.edges(g,t(edges))
        if(rem.multi){
            E(g)$weight <- count.multiple(g)
            g<-simplify(g, remove.multiple = TRUE, remove.loops = TRUE,edge.attr.
            comb = 'mean')
        }
        g
}
```

这里的参数 data 是包括点名称以及线数据的数据框，并且分为 from 列和 to 列。例如：

```
from<-c("a","a","e","b","b","c","d","d","d","f")
to<-c("c","e","c","e","c","d","g","g","f","d")
data<-data.frame(from=from,to=to)
```

参数 dir 是逻辑变量，设为 TRUE 时，表示建立有向图，否则是无向图。参数 rem.multi 是逻辑变量，设为 TRUE 时，表示删除重复变量并更新线权重 weigth，否则不删除并且线权重 weigth 恒为 1。

3. 基础概念的 R 语言实现

依次使用 init.igraph 函数建立无向图和有向图，并使用 plot 函数绘制图像。代码如下：

```
par(mfcol=c(1,2))
g.undir<-init.igraph(data)
plot(g.undir,edge.width=E(g.undir)$weight,main="无向图 g.undir",
 edge.label=E(g.undir)$weight)
g.dir<-init.igraph(data,dir=T)
plot(g.dir,edge.width=E(g.dir)$weight,main="有向图 g.dir", edge.label=E(g.dir)$weight)
```

绘制结果如图 9-10 所示。

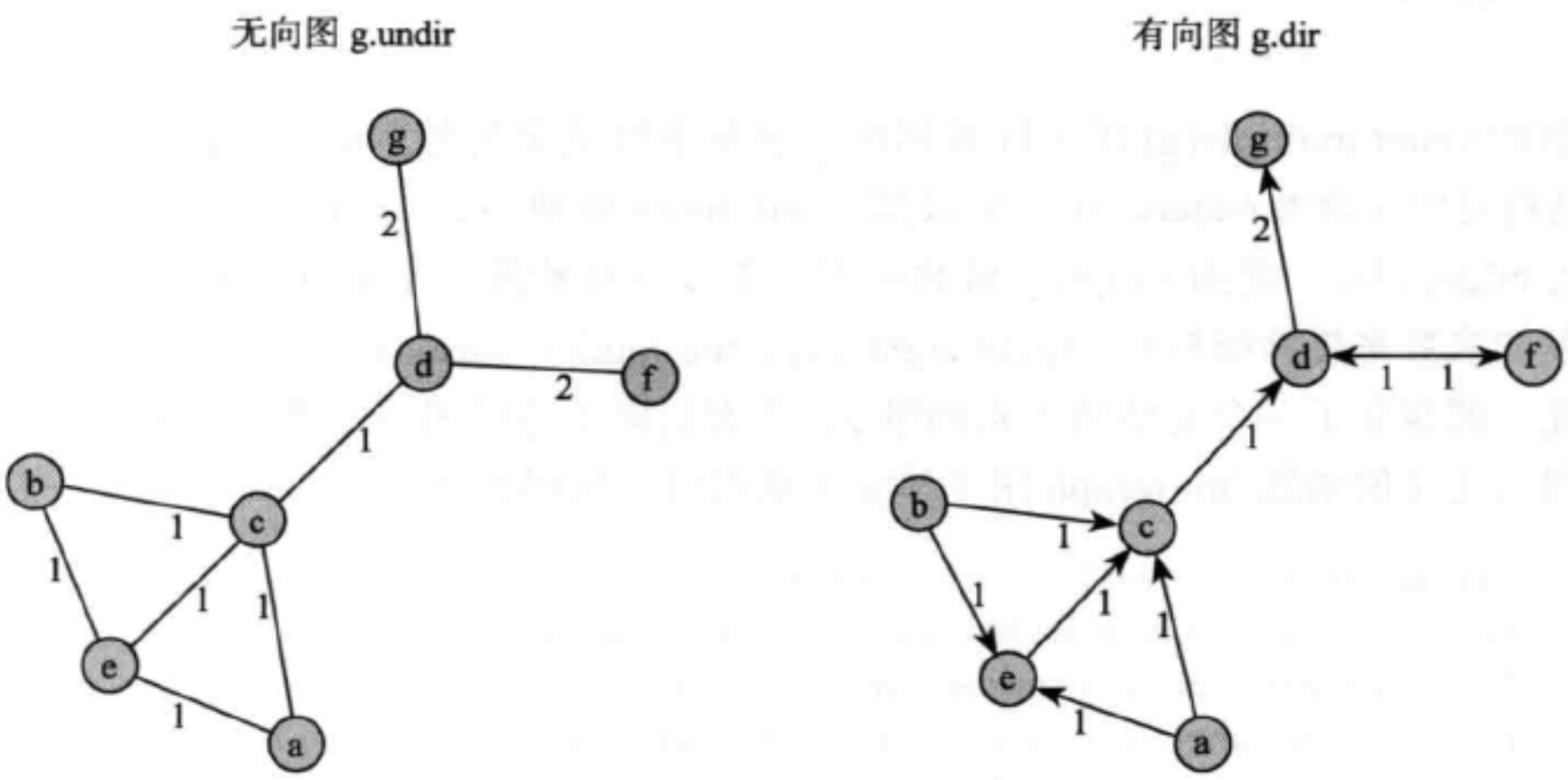


图 9-10 结果展示图

注意

这里通过设置 plot 函数的 edge.width 参数为 E(g)$weight，使得权重较高的线更粗。

下面使用 R 语言实现表 9-11 中的各个基础概念。

（1）点集合 (Vertexs) 和点的属性数据

使用 V(g) 可返回关系网络 g 中所有点的集合 V，并通过 length 函数直接返回点数目 n。代码如下：

```
V(g.undir)
Vertex sequence:
[1] 1 2 3 4 5 6 7
length(V(g.undir))
[1] 7
```

在 g.undir 中记录了点名称属性数据 V(g.undir)$label，这里也可以直接展示。代码如下：

```
V(g.undir)$label
[1] "a"  "e"  "b"  "c"  "d"  "f"  "g"
```

注意，在点数据 V(g.undir) 中可任意添加属性数据。例如为各点添加分组标签 membership，并分别为每组内的点分配一种颜色（color），相关代码如下：

```
member<-spinglass.community(g.undir)
V(g.undir)$membership<- member$membership; V(g.undir)$membership
[1] 2 2 2 2 1 1 1
mem.col<-rainbow(length(unique(V(g.undir)$ membership)),alpha=0.3)
V(g.undir)$color<-mem.col[V(g.undir)$ membership]; V(g.undir)$color
[1] "#00FFFF4C" "#00FFFF4C" "#00FFFF4C" "#00FFFF4C" "#FF00004C" "#FF00004C"
"#FF00004C"
    plot(g.undir,edge.width=E(g.undir)$weight, vertex.color=V(g.undir)$color)
```

绘制结果如图 9-11 所示。

其中，spinglass.community() 是一种发现社群的函数，返回对象中的 membership 列数据是每个样本点的分类组号，详细内容会在后续章节提到。

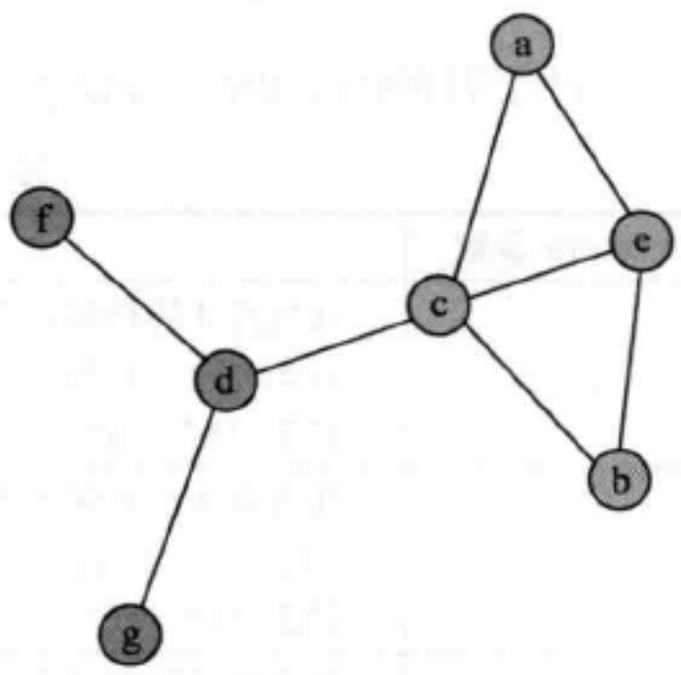


图 9-11 结果展示图

（2）点筛选和删除

点数据 V(g.undir) 的读取方式类似于读取向量，而点的属性数据 V(g.undir)$label 本身就是一个向量。例如，可以使用 V(g.undir)[1:2] 和 V(g.undir)$label[1:2] 来读取前 2 个样本点的内容。

```
V(g.undir)[1:2]
Vertex sequence:
[1] 1 2
V(g.undir)$label[1:2]
[1] "a" "e"
```

也可以通过 which 函数来筛选特定的点数据和点的属性数据。例如，读取第一组的点数据 V(g.undir) 以及点名称 V(g.undir)$label。代码如下：

```
V(g.undir)[which(V(g.undir)$ membership==1)]
Vertex sequence:
[1] 5 6 7
V(g.undir)$label[which(V(g.undir)$ membership==1)]
[1] "d" "f" "g"
```

上述代码也可以直接简写为：

```
V(g.undir)[membership==1]
Vertex sequence:
[1] 5 6 7
V(g.undir)[membership==1]$label
[1] "d" "f" "g"
```

使用减号（g- V(g)[i] 形式）可以从关系网络 g 中删除部分点 V(g)[i]，它是 add.vertices 函数的反操作。例如，删除关系网络 g.undir 中的孤立点（与任何其他点均没有关联），可使用如下代码：

```
g.undir<-g.undir- V(g.undir)[degree(g.undir)==0]
```

如果想删除第一组内的所有点数据，则使用如下代码：

```
g.undir <- g.undir-V(g.undir)[membership==1]
```

（3）相邻点集合

在 igraph 包中可以使用 neighbors 函数来找出某一点的相邻点的序号向量。例如，在无向图 g.undir 中找到和点 d 相邻的点以及该点名称。代码如下：

```
V(g.undir)[neighbors(g.undir, v= which(V(g.undir)$label=="d"))]
Vertex sequence:
[1] 4 6 7
V(g.undir)$label[neighbors(g.undir, v= which(V(g.undir)$label=="d"))]
[1] "c" "f" "g"
```

对于有向图 g.dir，可设置 mode 参数来指定 neighbors 的计算方式，如表 9-12 所示。

表 9-12 neighbors 函数的 mode 参数

mode 参数	代码实例
out	找出点 d 的相邻出点（点 d 为线的发起端：edges(d,f) 和 edges(d,g)），代码如下： `V(g.dir)$label[neighbors(g.dir, v= which(V(g.dir)$label=="d"),mode="out")]` `[1] "f" "g"`
in	找出点 d 的相邻入点（点 d 为线的接收端：edges(c,d) 和 edges(f,d)），代码如下： `V(g.dir)$label[neighbors(g.dir, v= which(V(g.dir)$label=="d"),mode="in")]` `[1] "c" "f"`
total	找出点 d 的相邻出点和入点的总和（点 d 为线的发起端或接收端：edges(d,f),edges(d,g), edges(c,d), edges(f,d)），代码如下： `V(g.dir)$label[neighbors(g.dir, v= which(V(g.dir)$label=="d"),mode="total")]` `[1] "c" "f" "f" "g"`

（4）点的度数

1）无向图的点度数。在无向图中，点的度数等于该点的相邻点数。例如，无向图 g.undir 中点 d 的度数为：

```
length(neighbors(g.undir, v= which(V(g.undir)$label=="d")))
[1] 3
```

在 igraph 包中也可以直接使用 degree 函数来计算。

```
#### 设置参数 v 方式 ####
degree(g.undir, v= which(V(g.undir)$label=="d"))
[1] 3
#### 向量方式 ####
```

```
degree(g.undir)[ which(V(g.undir)$label=="d")]
[1] 3
```

其中 degree(g.undir) 以向量形式直接返回关系网络 g.undir 内所有点的度数。

```
degree(g.undir)
[1] 2 3 2 4 3 1 1
```

2）有向图的点度数。对于有向图 g.dir，也可设置 mode 参数来指定 degree 函数的计算方式。以下代码分别计算点 d 的出度、入度以及综合度数。

```
#### 设置参数 v 方式 ####
degree(g.dir, v= which(V(g.dir)$label=="d"),mode="out")
[1] 2
degree(g.dir, v= which(V(g.dir)$label=="d"),mode="in")
[1] 2
degree(g.dir, v= which(V(g.dir)$label=="d"),mode="total")
[1] 4
#### 向量方式 ####
degree(g.dir,mode="out")[ which(V(g.dir)$label=="d")]
[1] 2
degree(g.dir, mode="in")[ which(V(g.dir)$label=="d")]
[1] 2
degree(g.dir, mode="total")[ which(V(g.dir)$label=="d")]
[1] 4
```

（5）线集合和线的属性数据

使用 E(g) 可返回关系网络 g 中所有线的集合 E(edges)，并通过 length 函数直接返回线数目 m，如表 9-13 所示。

表 9-13 线集合以及线数目

	无向图 g.undir	有向图 g.dir
线集合 E	E(g.undir) Edge sequence: [1] 2--1 [2] 4--1 [3] 3--2 [4] 4--2 [5] 4--3 [6] 5--4 [7] 6--5 [8] 7--5	E(g.dir) Edge sequence: [1] 1→2 [2] 1→4 [3] 2→4 [4] 3→2 [5] 3→4 [6] 4→5 [7] 5→6 [8] 5→7 [9] 6→5
线数目 m	length(E(g.undir)) [1] 8	length(E(g.dir)) [1] 9

线数目也可以直接添加属性数据，并使用属性作为筛选条件。例如在 9.3.1 节中添加的线权重属性 weight 为：

```
E(g)$weight <- count.multiple(g)
```

（6）线的筛选和删除

线的筛选和删除操作与点数据类似。例如，读取无向图 g.undir 中线权重大于 1 的线，代码如下：

```
E(g.undir)[(weight>1)]
Edge sequence:
[7] 6 -- 5
[8] 7 -- 5
```

也可以使用减号（g- E(g)[i] 形式）从关系网络 g 中删除部分线 E(g)[i] 的功能，它是 add.edges 函数的反操作。下面的代码从无向图 g.undir 中删除线权重大于 1 的线。

```
g.undir<- g.undir-E(g.undir)[(weight>1)]
```

（7）两点间的最短路径和距离

使用 igraph 包中的 get.shortest.paths 函数可读取两点间的最短路径，该函数的参数如表 9-14 所示。

表 9-14 get.shortest.paths 函数的参数

参数	说明
graph	关系网络对象，如图 9-10 中的 g.undir 和 g.dir
from	路径的起点，如果不设置参数 to，则计算读取点 from 到其他各点的最短路径。例如，读取点 a 与其他各点的最短路径为： get.shortest.paths(g.undir,from=V(g.undir)[label=="a"])
to	路径的终点，设置该参数后，计算读取点 from 到点 to 的最短路径。例如，读取点 a 到点 b 的最短路径为： get.shortest.paths(g.undir,from=V(g.undir)[label=="a"],to=V(g.undir)[label=="b"]) [[1]] [1] 1 4 3
mode	对于有向图，可以设置计算方式，其取值可为 "in"、"out"、"all"。例如： mode= "in" 时，表示点 c 到点 g 并无路径。 mode= "out" 时，表示点 c 到点 g 有路径 c → d → g。 mode= "all" 时，为上述情况总和，即有 1 条路径 c → d → g

另外，使用 igraph 包中的 shortest.paths 函数可以读取两点间的距离（即最短路径的长度），该函数的常用参数如表 9-15 所示。

表 9-15 shortest.paths 函数的常用参数

参数	说明
graph	关系网络对象，如图 9-10 中的 g.undir 和 g.dir，如果不设置 v 和 to，则计算各个点之间的距离。例如，无向图 g.undir 内各个点间的距离为： shortest.paths(g.undir) [,1][,2][,3][,4][,5][,6][,7] [1,] 0 1 2 1 2 4 4 [2,] 1 0 1 1 2 4 4 [3,] 2 1 0 1 2 4 4 [4,] 1 1 1 0 1 3 3 [5,] 2 2 2 1 0 2 2 [6,] 4 4 4 3 2 0 4 [7,] 4 4 4 3 2 4 0
v	路径的起点，如果不设置参数 to，则计算读取点 v 到其他各点的距离。例如，读取点 a 同其他各点的距离： shortest.paths(g.undir,v=V(g.undir)[label=="a"]) [,1][,2][,3][,4][,5][,6][,7] [1,] 0 1 2 1 2 4 4
to	路径的终点，设置该参数后，计算读取点 v 到点 to 的距离。例如读取点 a 和点 b 的距离： shortest.paths(g.undir,v=V(g.undir)[label=="a"],to=V(g.undir)[label=="b"]) [,1] [1,] 2 点 c 和点 f 的距离为： shortest.paths(g.undir,v=V(g.undir)[label=="c"],to=V(g.undir)[label=="f"])

（续）

参数	说明
to	[,1] [1,] 3 由于路径 d → f 中间有两条线（线权重为 2），所以最后的距离为：1（c 和 d 间距离）+2（d 和 f 间距离）=3
mode	对于有向图，可以设置计算方式，其取值可为 "in"，"out"，"all"，具体含义请参考 get.shortest.paths 函数

如果想把无向图 g.undir 中点 c 到点 g 之间的最短路径绘制为红线并加粗，且路径所经过的相关点设置为绿色，则代码如下：

```
pa<-get.shortest.paths(g.undir, from=which(V(g.undir)$label=="c"),
to=which(V(g.undir)$label=="g"))[[1]]
    E(g.undir)$color<-'black'  ; E(g.undir,path=pa)$color <- 'red'
    E(g.undir,path=pa)$width <- 3
    V(g.undir)[pa]$color <- 'green'
    plot(g.undir)
```

绘制结果如图 9-12 所示。

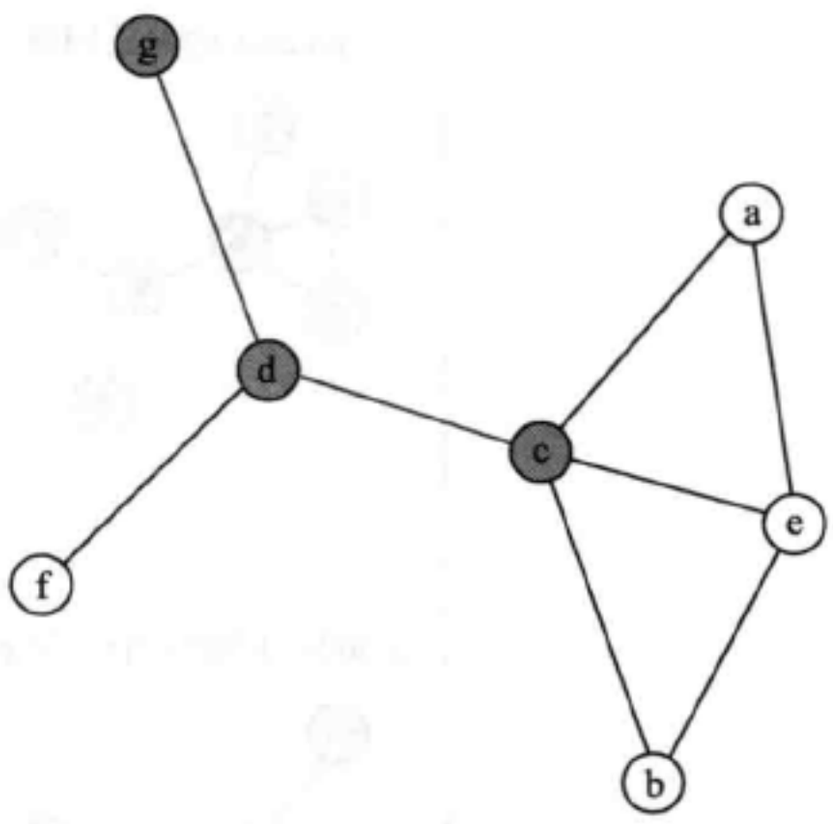


图 9-12 结果展示图

代码 E(g.undir,path=pa) 用于截取出经过 pa 各个点的相关线（通过设置 E 函数的 path 参数实现），结果如下：

```
pa
[1] 4 5 7
E(g.undir,path=pa)
Edge sequence:
[6] 5 -- 4
[8] 7 -- 5
```

（8）关系网络的筛选

使用 induced.subgraph 函数可以通过点筛选出关系网络。例如，要筛选出无向图 g.undir 中点 f、d、g、c 的图，可采用如下代码：

```
tmp.v<-c(which(V(g.undir)$label==c("c")), which(V(g.undir)$label==c("d")),
         which(V(g.undir)$label==c("g")), which(V(g.undir)$label==c("f")))
g.undir.1<- induced.subgraph(g.undir, tmp.v);
# 也可以使用: g.undir.1<- induced.subgraph(g.undir, V(g.undir)[tmp.v]);
```

其中，tmp.v 是预筛选的点的 idx 号，在 g.undir 中，点 f、d、g、c 实际是点 4、5、7、6。另外，语句 g.undir.1<- induced.subgraph(g.undir, tmp.v) 也可以使用如下语句代替：

```
g.undir.1<- induced.subgraph(g.undir, c(4,5,7,6));
g.undir.1<- induced.subgraph(g.undir, V(g.undir)[tmp.v]);
```

使用 subgraph.edges 函数通过连续筛选得到关系网络。例如，从无向图 g.undir 中筛选出连接点 c 和 g 之间的连线：

```
    pa<-get.shortest.paths(g.undir, from=which(V(g.undir)$label=="c"),
to=which(V(g.undir)$label=="g"))[[1]]
    g.undir.2<-subgraph.edges(g.undir, E(g.undir,path=pa))
```

或者筛选出线权重大于1的线（点d、f、g之间的连线）：

```
    g.undir.3<-subgraph.edges(g.undir, which(E(g.undir)$weight>1))
```

最后使用如下语句展示筛选后的网络图。

```
    par(mfcol=c(2,2))
    plot(g.undir, edge.width=E(g.undir)$weight,main="g.undir 的关系网络 ")
    plot(g.undir.1, edge.width=E(g.undir.1)$weight,main="g.undir 中筛选出点 f,d,g,c")
    plot(g.undir.2, edge.width=E(g.undir.2)$weight,main="g.undir 中筛选点 c 和 g 之间的连线 ")
    plot(g.undir.3, edge.width=E(g.undir.3)$weight,main="g.undir 中筛选线权重大于 1 的连线 ")
```

绘制结果如图9-13所示。

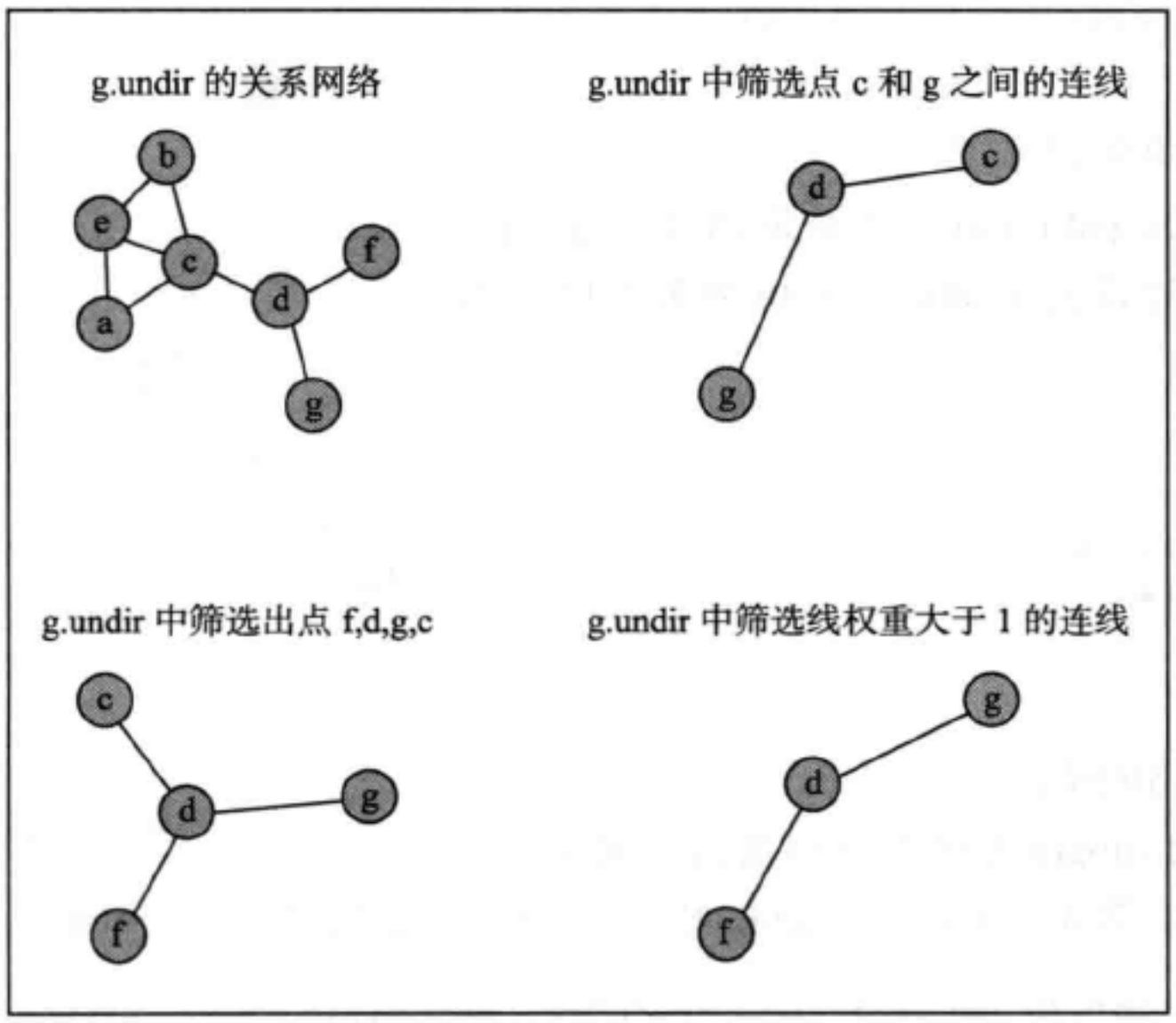


图9-13 结果展示

9.3.2 中心度

上面介绍了关系网络的基本概念，以及如何使用igraph包来建立并操作一个关系网络，下面引入一系列的中心度概念，用于衡量某一点在整个关系网络中的“核心地位”。

1. 点度中心度

（1）绝对点度中心度

如果一个点与很多点有直接联系，即某条信息被该点传播后可以达到更多的点，则可以

认为该点居于核心地位。相反如果一个点只与少数几个点有直接联系，则认为该点居于边缘地位。一个点与其他点的直接联系程度是使用“度”来衡量的，所以这里引入点度中心度（degree）的概念。

无向图的点度中心度公式如下：

点度中心度 = 点度数：与该点相连的点的总数

有向图的点度中心度公式如下：

内点度中心度 = 点入度：直接指向该点的点的总数

外点度中心度 = 点出度：该点所直接指向的点的总数

点度中心度 = 点入度 + 点出度

使用上面的公式作为点度中心度的定义式其实有一个局限：中心度数值只有在同一个图或同等规模的图内才有比较的意义，同时它还强烈地依赖于图的规模，因此这种情况也称为绝对点度中心度（记为 C_D）。例如，通过 1000 个用户间留言记录的数据集，计算出用户 a 的点度中心度。另外，通过只有 10 个用户间留言记录的数据集，计算出用户 b 的点度中心度。在这种情况下比较用户 a 和用户 b 的点度中心度大小是毫无意义的。

在 R 语言中，直接使用 degree 函数就可以计算绝对点度中心度，具体参见 9.3.1 节。

（2）相对点度中心度

为了克服绝对点度中心度的这一缺点，在点度中心度的计算中引入了关系网络中点最大可能的度数，并称其为相对点度中心度（记为 C_D^1）。相对点度中心度 = 绝对点度中心度 / 关系网络中点的最大可能度数。

无向图的相对点度中心度公式如下：

点度中心度 = 点度数 /(n−1)

有向图的相对点度中心度公式如下：

内点度中心度 = 点入度 /(n−1)。degree 函数的 mode 参数为 in。

外点度中心度 = 点出度 /(n−1)。degree 函数的 mode 参数为 out。

点度中心度 =(点入度 + 点出度)/(2n−2)。degree 函数的 mode 参数为 total。

其中 n 为关系网络 graph 中的点数 length(V(graph))。使用相对点度中心度，可以直接对比不同网络中各点的点度中心度。

在 R 语言中，可以设置 degree 函数的 normalized 参数为 TRUE 来计算相对点度中心度。

注意

点度中心度所考虑的问题是，一个点在局部环境内与其他点的直接关联，因为它并没有考虑到间接关联的点，所以测量出来的中心度只是局部中心度。

2. 接近中心度

如果一个点可以更快（通过很少的点做中介）地到达其他各点，即该点所发出的信息可

以更快地传播到整个网络，则可以认为该点居于核心地位。由这一思路可引入点的接近中心度（closeness）概念（记为 C_G），它等于该点与网络中其他点距离之和的倒数。当两点间距离无限大（INF）时（两点间没有相连接），使用网络中点的总数来代替。

（1）无向图的接近中心度

例如本节引例中的无向图 g.undir 中点 c 与其他各点的距离如表 9-16 所示。

表 9-16　无向图的接近中心度

	a	b	d	e	f	g
与点 c 的距离	1	1	1	1	3	3

注意，由于 d 和 f 两点间的线权重为 2，所以 c 和 f 的间距为 1+2=3，同理 c 和 g 的间距也为 3。因此，点 c 的接近中心度 C_G 为 1/(1+1+1+1+3+3)=0.1。

可以使用如下 shortest.paths 函数来计算接近中心度。

```
pa<-shortest.paths(g.undir,v=V(g.undir)[label=="c"])[-which(V(g.
undir)$label=="c")]
pa<-ifelse(pa==Inf,length(V(g.undir)),pa)
1/sum(pa)
```

首先计算点 c 与其他点的距离 pa，然后使用 ifelse 函数把距离为无限大 Inf 的数据（本例中暂未出现无限大）替换为点总数 length(V(g.undir))，最后取距离总和的倒数 1/sum(pa)。

在 igraph 包中也可以使用 closeness() 函数直接计算接近中心度：

```
closeness(g.undir,vids=which(V(g.undir)$label=="c"))
[1] 0.1
```

（2）有向图的接近中心度

例如，在本节引例中，有向图 g.dir 的点 c 与其他各点的出度距离（c 为发送端）如表 9-17 所示。

表 9-17　有向图的出度距离

	a	b	d	e	f	g
与点 c 的距离	INF	INF	1	INF	2	3

注意，点 a、b 和 e 与点 c 并无连接，即距离为 inf 无限大，此时用网络中点的总数来代替。点 c 的接近中心度 C_G 为 1/(7+7+1+7+2+3)= 0.03703704。通过设置 closeness() 函数的 mode 参数为 out 可以计算接近中心度。代码如下：

```
closeness(g.dir,vids=which(V(g.dir)$label=="c"),mode="out")
[1] 0.03703704
```

而在有向图 g.dir 中，点 c 与其他各点的入度距离（c 为接收端）如表 9-18 所示。

表 9-18　有向图的入度距离

	a	b	d	e	f	g
与点 c 的距离	1	1	INF	1	INF	INF

仍使用点数 7 来代替无限大 inf，则点 c 的接近中心度 C_G 为 1/(1+1+7+1+7+7)= 0.04166667。通过设置 closeness() 函数的 mode 参数为 in 可以计算接近中心度。代码如下：

```
closeness(g.dir,vids=which(V(g.dir)$label=="c"),mode="in")
[1] 0.04166667
```

（3）相对接近中心度

上面介绍的接近中心度实际是绝对接近中心度（记为 C_G）。和绝对点度中心度类似，绝对接近中心度也有受网络规模影响的缺点。这里定义的点的相对接近中心度（记为 C'_G），等于绝对接近中心度乘以 n−1（n−1 实际是在以 n 个点组成的网络里接近中心度的最大可能数值），即：

$C'_G = C^*_G(n-1)$

其中 n 是关系图 graph 的点数 length(V(graph))。使用相对接近中心度，可以直接对比不同网络中各点的接近中心度。

在 R 语言中，可以设置 closeness 函数的 normalized 参数为 TRUE 来计算相对点度中心度。

```
closeness(g.dir,vids=which(V(g.dir)$label=="c"),mode="in",normalized=T)
[1] 0.25
```

3. 中间中心度（betweenness）

（1）点的中间中心度

如果在网络中两点间的最短路径大多都经过某一个点，即信息的传播大多要经过该点才能完成，则可以认为该点处于核心地位。它控制了两个点之间的间接关联，即起到桥梁的作用。例如，使用如下代码：

```
g<-graph.tree(40,3,mode="undirected")
member<-spinglass.community(g,spins=3)
V(g)$member<- member$membership
mem.col<-rainbow(3,alpha=0.3)
V(g)$color<-mem.col[member$membership]
V(g)[1]$color<-"red"
plot(g, vertex.color=V(g)$color)
```

产生的图形如图 9-14 所示。

该图整体上可分为由点 2、3、4 逐步发散得到的三大部分（依次是粉色、蓝色、绿色 3 类点），并由点 1（红色点）统一连接。不同部分的两个点间的最短路径均要通过点 1，可以说点 1 对信息在该网络上的传播起到了很重要的中介作用。

假设点 j 和点 k 之间存在的最短路径数记为 g_{jk}，另一个点 i 控制这两个点联通的能力，用 $b_{jk}(i)$ 表示，即 $b_{jk}(i)$ 等于点 i 处于点 j 和点 k 间

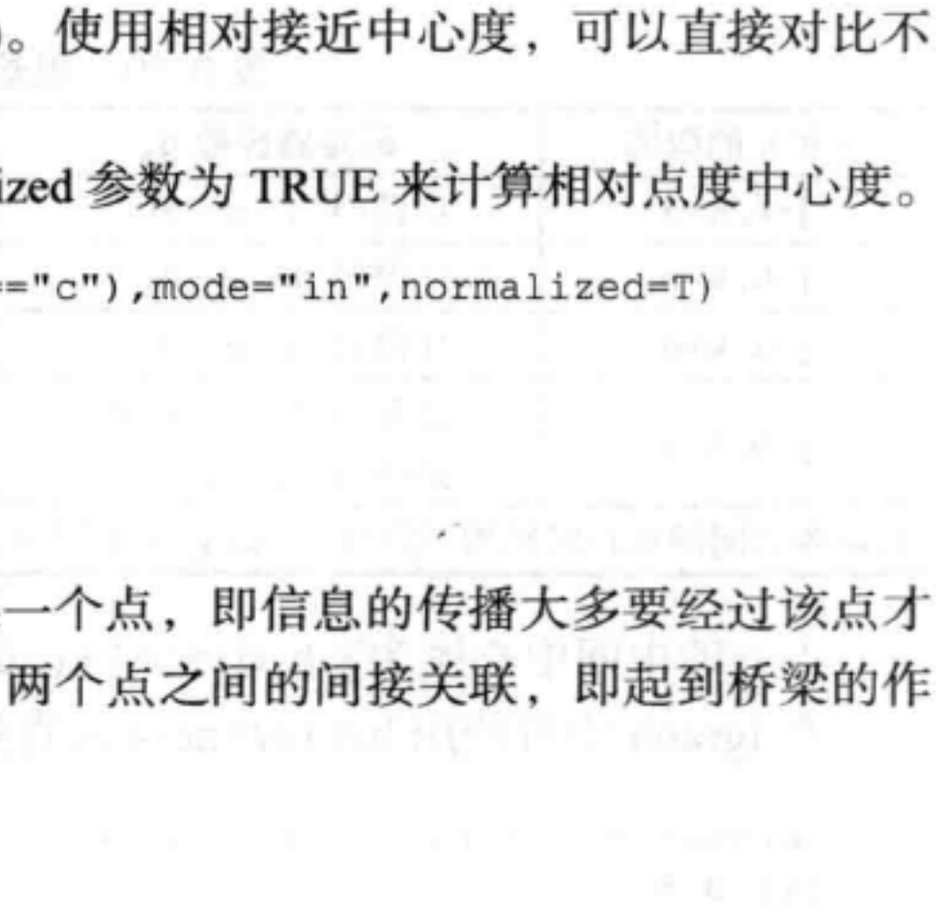


图 9-14 示例图

最短路径的概率。如果点 j 和点 k 间经过点 i 的最短路径数记为 $g_{jk}(i)$，则 $b_{jk}(i)= g_{jk}(i)/ g_{jk}$。那么，把所有 $b_{jk}(i)$ 的总和$\sum b_{jk}(i)$ 称为中间中心度。

现在把本节引例中无向图 g.undir 的点 g 和 f 删除，以简化案例便于讲解：

```
g<-g.undir-V(g.undir)[c(which(V(g.undir)$label=="f"),which(V(g.
undir)$label=="g"))]
plot(g)
```

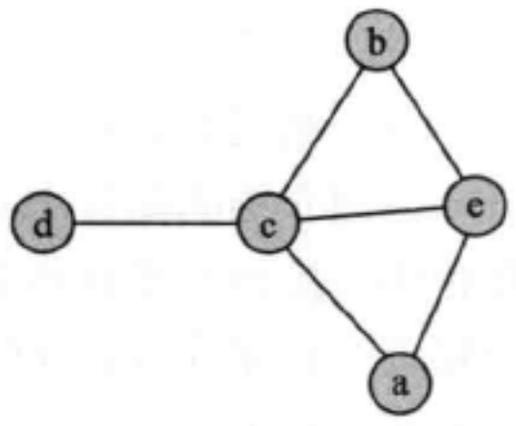


图 9-15 化简后的示例图

简化后的图示如图 9-15 所示。

现在欲研究 c 点的中间中心度，即 i 取 c，其中所涉及的最短路径和绝对中间中心度如表 9-19 所示。

表 9-19 最短路径和绝对中间中心度

j 和 k 的取值	最短路径数 g_{jk}	经过点 i 的最短路径数 $g_{jk}(i)$	概率 $b_{jk}(i)= g_{jk}(i)/g_{jk}$
j=e, k=d	1(路径 :e—c—d)	1(路径 :e—c—d)	1/1=1
j=b, k=d	1(路径 :b—c—d)	1(路径 :b—c—d)	1/1=1
j=a, k=d	1(路径 :a—c—d)	1(路径 :a—c—d)	1/1=1
j=b, k=a	2(路径 :b—c—a 和 路径 :b—e—a)	1(路径 :b—c—a)	1/2=0.5
其余各点间的最短路径均不经过 c，即 $g_{jk}(i)$ 恒为 0，所以概率 $b_{jk}(i)$ 也为 0			

点 c 的中间中心度为$\sum b_{jk}(i)$=1+1+1+0.5=3.5。

在 igraph 中可使用 betweenness 来直接计算点 C 的中间中心度：

```
betweenness(g,v=V(g)[which(V(g)$label=="c")])
[1] 3.5
```

对于有向图，也可以设置 mode 参数以不同方式计算，可以参考点度中心度和接近中心度的实例，这里不再深入讲解。

（2）点的相对中间中心度

上面介绍的中间中心度实际是绝对中间中心度（记为 C_B）。和绝对点度中心度类似，相对中间中心度也有受网络规模影响的缺点。这里定义点的相对中间中心度（记为 C'_B），它等于绝对中间中心度除以 $(n^2-3n+2)/2$，其中（$n^2-3n+2)/2$ 实际等于以 n 个点组成的网络里中间中心度的最大可能数值），即：

$C'_B= 2*C_B/(n^2-3n+2)$

其中 n 是关系图 graph 的点数 length(V(graph))。使用相对中间中心度，可以直接对比不同网络中各点的中间中心度。

在 R 语言中，可以设置 betweenness 函数的 normalized 参数为 TRUE 来计算相对点度中心度。

```
betweenness(g,v=V(g)[which(V(g)$label=="c")],normalized=T)
[1] 0.5833333
```

（3）线的中间中心度

线的中间中心度与点的中间中心度类似，这里不再详细介绍。在 igraph 包中，使用 edge.betweenness 函数来计算线的（绝对）中间中心度，例如，计算 b—c 线的中间中心度，可以使用如下代码：

```
    edge.betweenness(g, e=E(g,path= c(which(V(g.undir)$label=="b"),which(V(g.
undir)$label=="c"))))
    [1] 2.5
```

另外，各种中心度的区别如表 9-20 所示。

表 9-20 中心度的区别

中心度	意义	何时应用
点度中心度	描述的是行动者的局部中心指数。如果某点的中心度较高，则信息被该点传播后可以达到更多的点	研究信息交流活动
接近中心度	考虑的是行动者在多大程度上与其他行动者相互影响。如果某点的中心度较高，则该点所发出的信息可以更快地传播到整个网络中	研究信息传递的有效性和独立性
中间中心度	研究一个行动者在多大程度上居于其他两个行动者之间，因而是一种“控制能力”指数。如果某点的中心度较高，则信息的传播大多要经过该点才能完成	研究对信息交流的控制

4. 其他中心度度量

（1）特征向量中心度

在一个社会群体中，如果某用户的朋友大多是一些重要人物，则认为该用户也是重要人物。对于特征向量中心度（eigenvector，记为 C_E）的具体数学原理，这里不展开讲解，读者只要知道它是依据某点（点 v_i）相邻点（点 $v_j \in Ni$）的重要性来衡量该点重要性的一种度量方式即可，即：

$C_E(v_i) \propto \sum A_{ij} * C_E(v_j)$

其中 A_{ij} 是邻接矩阵，如果点 v_i 和点 v_j 之间有关联，则 a_{ij} 为 1，否则 a_{ij} 为 0。注意，即使只是计算一个点的特征向量 $C_E(v_i)$，系统也要把其他各点的数值全部计算出来。

在 igraph 包中，使用 evcent 函数来计算网络中各点的特征向量中心度。例如：

```
evcent(g.undir,scale=F)$vector
[1] 0.2101088   0.2603852   0.2101088   0.4075667   0.6150825   0.3869568   0.3869568
```

上面代码计算了无向图 g.undir 中各个点的特征向量中心度。如果把 scale 设为 TRUE（默认就为 TRUE），则系统返回的结果会默认归一化，从而使结果的范围为 [0,1]，相关代码如下：

```
evcent(g.undir)$vector
[1] 0.3415946   0.4233338   0.3415946   0.6626212   1.0000000   0.6291137   0.6291137
```

（2）page.rank 中心度

在特征向量中心度的基础上可以引申出 google 的 page.rank 中心度。首先把邻接矩阵 A_{ij} 依

据列规范化后得到新的邻接矩阵 A_{ij}'，然后使用类似于特征向量中心度的方式计算出 page.rank 中心度。具体的计算原理这里不再详细介绍，在 igraph 包中，使用 page.rank 函数来实现。

```
page.rank(g.undir)$vector
[1] 0.1064008 0.1533830 0.1064008 0.1953587 0.2354760 0.1014904
0.1014904
```

9.3.3 中心势

上面介绍了如何描述一个点在整个网络中的中心地位，有时比较关心的是整个网络向某一区域的集中趋势。例如，一个星形关系网络的集中趋势会比其他形态的网络要高，这在对比不同网络或社区的性质时有重要的意义。

一般而言，一个具有较高集中趋势的网络，其中心点的中心度较高，边缘点较低，即各个点的中心度有较大的差异。而对于集中趋势不明显的关系网络，各个点的中心度差异性较小。基于这样的指导思想来定义中心势，其公式如下：

$$\text{中心势} = \frac{\sum^{n}(C_{\max}-C_i)}{\max[\sum^{n}(C_{\max}-C_j)]}$$

其中 n 为网络中的点数。中心势的含义是：首先计算出网络中的最大中心度数值 $C_{\max}$，然后计算 $C_{\max}$ 与其他任何点中心度 C_i 的差值，并求和$\sum^{n}(C_{\max}-C_j)$。最后用该总和除以各个差值总和的最大可能值（此时网络是一个包含 n 个点的星形网络结构）。

在实际应用中，可以使用点度中心度、接近中心度以及中间中心度来代替点中心度 C_i，具体的中心势定义公式见表 9-21。

表 9-21 中心势定义

C_i 的取值		中心势定义式
点度中心度	绝对点度中心度 C_D	$\frac{\sum^{n}(C_{\max}-C_i)}{n^2-3n+2}$
	相对点度中心度 C'_D	$\frac{\sum^{n}(C_{\max}-C_j)}{n-2}$
接近中心度	绝对接近中心度 C_G	$\frac{\sum^{n}(C_{\max}-C_i)}{(n-1)(n-2)} *(2n-3)$
	相对接近中心度 C'_G	$\frac{\sum^{n}(C_{\max}-C_i)}{(n-2)} *(2n-3)$
中间中心度	绝对中间中心度 C_B	$\frac{\sum^{n}(C_{\max}-C_i)}{n^3-4n^2+5n-2}$
	相对中间中心度 C'_B	$\frac{\sum^{n}(C_{\max}-C_i)}{n-1}$

9.3.4 社群发现

社群的结构特点为：社群内的边密度要高于社群间的边密度，即社群内部节点之间的连接相对紧密，各社群之间的连接相对比较稀疏。

1. 模块化指标

（1）模块化指标 Q 定义

模块化指标 Q 可用于衡量各个点社群化的程度，其核心思想是希望相同社群内点之间的关联尽量多，而不同社群间点之间的关联尽量少，公式如下：

$$Q=\sum_{i=1}^{m}(e_{ii}-(\sum_{j=1}^{m}e_{ij})^2)=\sum_{i=1}^{m}(e_{ii}-a_i^2)$$

其中，m 为整个网络中所有线的（关系）数目。e_{ii} 为社群 i 内所有线的数目 $/m$。e_{ij} 为社群 i 内所有点的度数 $/2m$。例如，图 9-16 中有 A ～ D 共 4 个点，其中 A 和 B 之间有连线，C 和 D 之间有连线。现在要计算当把 A、B、D 三点作为一个社群，C 单独作为一个社群时的 Q 值，计算结果如表 9-22 所示。

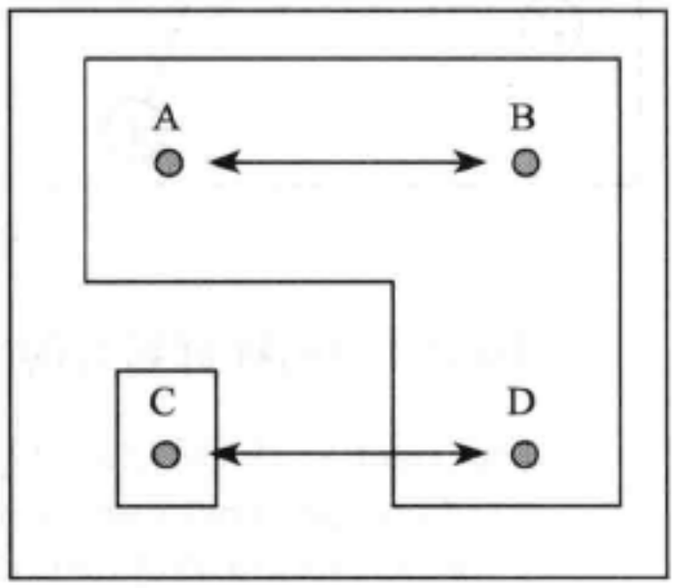


图 9-16 示例图

表 9-22 Q 值的计算示例

社群	e_{ii}	a_i	a_i^2	e_{ii}-a_i^2
1(包含点 A、B、D)	1/2	(1+1+1)/(2*2)	9/16	-1/16
2(包含点 C)	0/2	1/(2*2)	1/16	-1/16
Q				-1/8

（2）模块化指标的 R 语言实现

在 igraph 包中，使用 modularity 函数来计算 Q 值。例如，把无向图 g.undir 的点 1 ～ 3 合为社群 1，点 4 ～ 7 合为社群 2 时计算 Q 值，代码如下：

```
modularity(g.undir,membership=c(1,1,1,2,2,2,2))
[1] 0.1171875
```

其中 membership 参数用于设置各点所在的社群号，而社群发现的目的就是获取社群号向量。

2. 基于点连接的社群发现

最简单的社群发现方式是：如果一个点和某社群之间有关联（有线连接），则把该点归于该社群中。社群发现的具体规则如表 9-23 所示。

可使用 clusters 函数来发现该类社群。例如，无向图 g.undir 以及有向图 g.dir 的社群划分（相同社群内的点，其颜色相同）如图 9-17 所示。

表 9-23 社群发现规则

类型		规则
无向图		两点间有连线，则视为一个社群
有向图	弱关联	两点间有单向连线，则视为一个社群
	强关联	两点间有双向连线，则视为一个社群

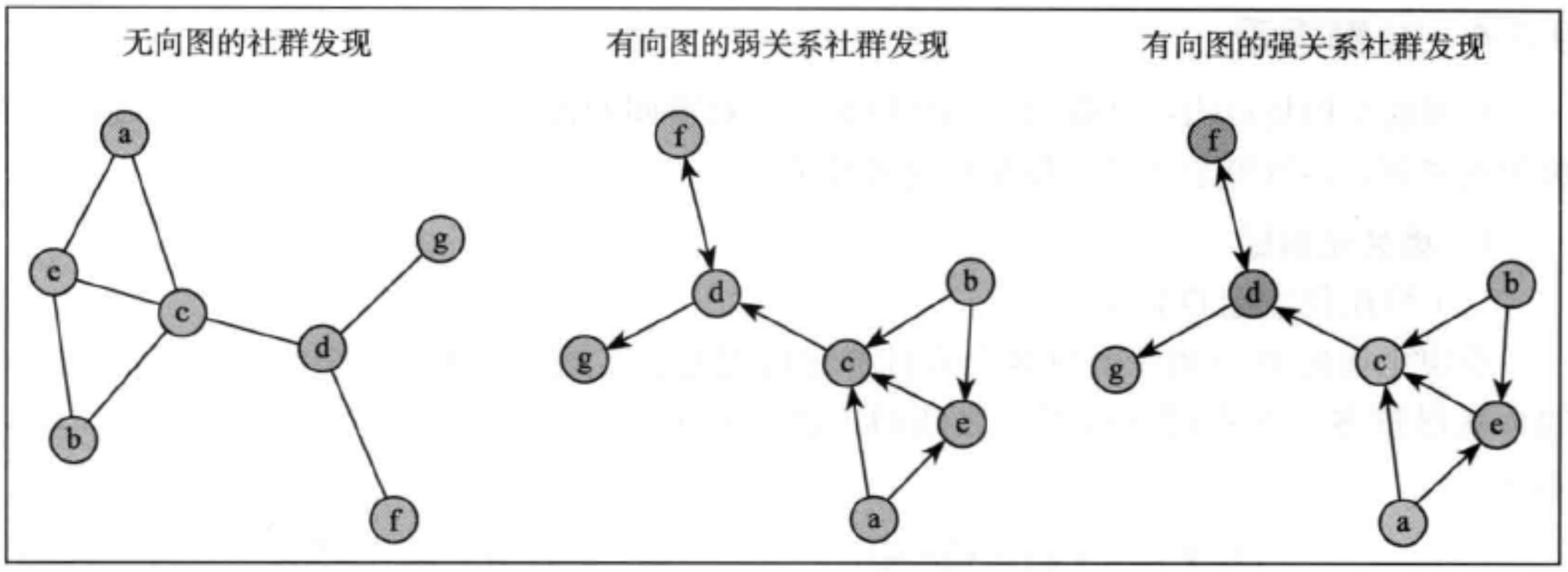


图 9-17 基于点连接的社群发现示例图

图 9 -17 中的社群划分的实现代码如下：

```
plot.membership<-function(graph,membership,main=""){
    V(graph)$member<-membership
    mem.col<-rainbow(length(unique(membership)),alpha=0.3)
    V(graph)$color<-mem.col[membership]
    plot(graph,edge.width=E(graph)$weight, vertex.color=V(graph)$color,main=main)
}
par(mfcol=c(1,3))
plot.membership(g.undir,clusters(g.undir)$membership," 无向图的社群发现 ")
plot.membership(g.dir,clusters(g.dir,mode="weak")$membership," 有向图的弱关系社群发现 ")
plot.membership(g.dir,clusters(g.dir,mode="strong")$membership," 有向图的强关系社群发现 ")
```

其中，自定义函数 plot.membership 用于把相同社群内的点置为同一种颜色，其中 graph 参数是关系网络对象，参数 membership 是各点的社群分组，参数 main 是绘图的主标题设定。

使用 no.clusters 函数可以计算社群数。例如，有向图 g.dir 的强关系社群数为 6，相关代码如下：

```
no.clusters(g.dir,mode="strong")
[1] 6
```

3. 随机游走社群发现

随机游走是由一连串轨迹组成的数学统计模型，其中每一次轨迹的移动都是基于 [0,1] 区间均匀随机分布的。它能用来表示不规则的变动形式，最典型的实例是布朗运动。随机游走社群发现的思想是假设一个漫步者在一个关系网络中随机游走，该漫步者很可能被困在连接稠密的区域里，这个稠密区域就是漫步者发现的社群。如果根据点和社群之间的结构相似度来度量漫步者的“行为”，那么这种相似度是一种可计算的、高效的距离相似度，这样就可以利用合并层次聚类方法建立社区的层次结构，也就是社区结构。这里可以利用从 i 点走向 j 点的概率定义距离相似度，若 i 和 j 在同一社群里，则概率相对比较高。

在 igraph 包中，可以使用 walktrap.community 函数来实现随机游走的社群发现，具体参

数如表 9-24 所示。

表 9-24 walktrap.community 函数的参数

参数	说明
graph	关系网络
weights	线权重，默认为 E(graph)$weight
steps	随机游走的步数，默认为 4。该值越大，社群划分得越粗糙 (数目越少)

下面的代码使用随机游走对无向网络 g.undir 进行社群发现，并为每组设定一个颜色，同时通过 plot 函数的 mark.groups 参数设置点的分组。

```
member<-walktrap.community(g.undir)
V(g.undir)$member<- member$membership
mem.col<-rainbow(3,alpha=0.3)
V(g.undir)$color<-mem.col[member$membership]
member.num<-length(table(V(g.undir)$member))
member.list<-list()
for(i in 1:member.num){
    member.list<-c(member.list,list(which(V(g.undir)$member==i)))
}
plot(g.undir, vertex.color=V(g.undir)$color, mark.groups=member.list)
```

展示结果如图 9-18 所示。

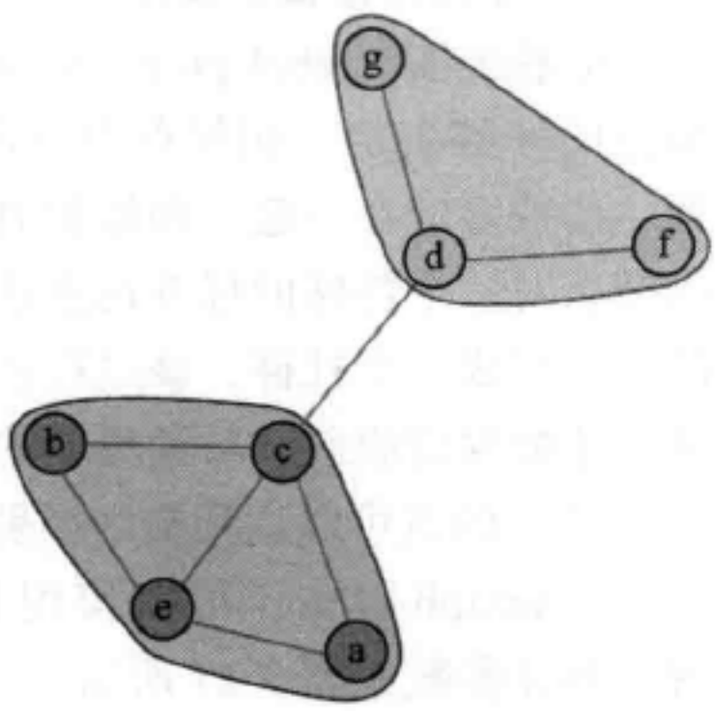


图 9-18 随机游走社群发现示例图

在返回的社群信息对象 member 中，member 属性记录各个点的社群编号。modularity 属性记录不同分割次数后的模块化指标，其中的最大值 max(member$modularity) 就是最后的模块化指标 Q。

4. 自旋玻璃社群发现

自旋玻璃源于物理学，它是某些特制的物理材料的一种无序状态。如果把关系网络看成一个随机网络场，网络中的某一个点与其他点的连接和不连接，可看作磁性材料的磁性相互作用和反磁性相互作用，两种相互作用形成一个能力函数，当能量函数最小时，关系网络的层次结构被解释为旋转配置，此时网络内部的社群可看作自旋系统的状态，该过程类似于层次聚类过程。

在 igraph 包中，可以使用 spinglass.community 函数来实现随机游走的社群发现，具体参数如表 9-25 所示。

表 9-25 spinglass.community 函数的参数

参数	说明
graph	关系网络
weights	线权重，默认为 null
spins	产生的社群数，默认为 25

下面的代码使用自旋玻璃对无向网络 g.undir 进行社群发现，并设置期望的社群数为 2。

```
member<-spinglass.community(g.undir,
weights = E(g.undir)$weight,spins=2)
```

```
V(g.undir)$member<- member$membership; V(g.undir)$member
[1] 1 1 1 1 2 2 2
```

注意，与随机游走不同，这里要设置参数 weights，因为 spinglass.community 函数的 weights 参数无默认值。

5. 基于中间中心度的社群发现

如果找出关系网络中相互关联最弱的点（使用中间中心度来找到），删除它们之间的边，并以此为分裂法对关系网络进行逐层划分（类似于层次聚类），那么就可以得到越来越小的社群。在适当的时刻终止这个分裂过程，得到最终的社群划分结果。

在 igraph 包中，可以使用 edge.betweenness.community 函数来实现随机游走的社群发现，具体参数如表 9-26 所示。

表 9-26 edge.betweenness.community 函数的参数

参数	说明
graph	关系网络
weights	线权重，默认为 E(graph)$weight
directed	只对有向图起作用，如果为真，则考虑有向连接，否则不考虑

下面的代码使用基于中间中心度的算法对无向网络 g.undir 进行社群发现。

```
member<-edge.betweenness.community (g.undir)
```

6. 标签传播社群发现

标签传播（label propagation）算法的基本思想是：每个点的社群归属（即标签）由相邻节点的标签决定，相邻点中出现最多的那个标签即为该节点的标签；可以理解为由邻居投票，票数多的在一起。初始化时，每个点都有一个独特的标签，通过标签的传播，密集连接的节点对一个特殊的标签迅速达成共识，这些密集的节点迅速扩张，直至这些具有相同标签的节点组成一个社群，该过程通过迭代完成。但是，此算法不保证收敛，需要添加一些限定条件才能保证收敛，一种简单的方法是迭代几次之后随机地断开。一般而言，通过 5 次迭代后，95% 的点可以达到最终类别。或者在初始化时，事先标注一些社群的中心节点。

在 igraph 包中，可以使用 label.propagation.community 函数来实现随机游走的社群发现，具体参数如表 9-27 所示。

表 9-27 label.propagation.community 函数的参数

参数	说明
graph	关系网络
weights	线权重，默认为 null
initial	可以对各个点做标签的初始化设置。默认为 null，表示初始化时，每个点的标签均不同。否则输入一个数值向量，非负数表示对应的点有特定标签，负数表示忽略不设置特定标签
fixed	逻辑向量，如果为 TRUE，则对应点的标签依据初始化设置固定不变，否则不固定可以变化。对于未设置初始化的点（initial=NULL 或向量的点为负数），该参数无效，设置为 F 即可

下面的代码使用标签传播对无向网络 g.undir 进行社群发现。

```
member<- label.propagation.community (g.undir,weights = V(g.undir)$weight)
```

```
    V(g.undir)$member
    [1] 1 1 1 1 1 1 1
    member<- label.propagation.community (g.undir,weights = E(g.undir)$weight,initi
al=c(1,1,-1,-1,2,-1,1))
    V(g.undir)$member
    [1] 1 1 1 1 2 2 2
```

当不设置初始化参数 initial 时，所有的 7 个点均为一个社群（标签为 1）。当设置 initial 为 c(1,1,−1,−1,2,−1,1) 时，表示把点 1、2、7 初始化为一个社群（initial 设置为 1），点 5 为一个社群（initial 设置为 2），点 3、4、5 不做初始化处理（initial 设置为 −1）。最后得到的结果为点 1 ～ 4 是一个社群，5 ～ 7 为一个社群。注意，虽然点 7 在初始化时是同点 1 和 2 在一组的，但是最后变为了与点 5 为一组。如果想把点 7 始终固定为原有社群标签不变，则需要设置 fixed 参数。代码如下：

```
    member<- label.propagation.community (g.undir,weights = E(g.undir)$weight,initi
al=c(1,1,-1,-1,2,-1,1),
                fixed=c(T,F,F,F,F,F,T))
    V(g.undir)$member
    [1] 1 1 1 1 1 1 1
```

由于点 3、4、5 不做初始化处理（initial 取值是 -1），所以 fixed 对应的向量元素只能为 F（设置为 T 也无效）。固定点 1 和点 7 的初始化社群信息（设置为 T），对于点 2 和 5 不固定（设置为 F），最后设置 fixed 为 c(T,F,F,F,F,F,T)。这样处理后，点 7 不再改变社群（仍和点 1 和点 2 在一起），而点 5 被强制转化为社群 1。

9.3.5 实例：分析微博的传播特性

下面使用某一条微博的发布 / 转发数据作为数据集来研究微博的传播特性。

1. 数据集

该数据集存储于“某一条微博传播数据 .csv”文件中，其格式如图 9-19 所示（使用 Excel 打开后的截图）。

csv 源数据文件包含 2 列，第 2 列表示发出（转发）该条微博的用户，第 1 列表示接收该条微博的用户。例如，由第一行可知，名字叫“老牛”的用户把该条微博发送 / 转发给了名字叫“晴”的用户。读入源数据文件，代码如下：



I23

	A	B	C
1	晴	老牛	
2	四眼看八方	老牛	
3	gaoqing	vuyue	
4	有效期2012	我是谁	
5	差不多先生avi	差不多先生avi	
6	话剧-人生	有餘	
7	我是八路	熊睿	
8	霉锁潇湘	熊睿	
9	史程博	熊睿	
10	易彬	赖波	
11	易彬	赖波	

图 9-19 数据集示例图

```
root="C:/Users/liming/Desktop/ 写书 /chap9/sna_igraph/"
init.data<-read.csv(file=paste(root," 某一条微博传播数据 .csv ",sep=""),header =F);
```

2. 建立关系网络和社群发现

（1）建立有向网络图 g

使用 9.3.1 节中自定义的 init.igraph 函数来建立一个有向图 g，代码如下：

```
data<-data.frame(from=init.data[,2],to=init.data[,1])
g<-init.igraph(data,dir=T)
```

由于读出数据 init.data 的第一列 init.data[,1] 是 csv 文件的首列数据，其含义是“接收该条微博的用户”，相当于有向网络图中的终点，所以标记为 to。同理，init.data[,2] 是 csv 文件的第二列数据，其含义是“发起 / 转发该条微博的用户”，相当于有向网络图中的起点，所以标记为 from。

（2）社群发现

这里使用随机游走来对该关系网络进行社群发现，得到 member 对象，并把每个点所属的社群号作为关系网络 g 的 member 属性。代码如下：

```
member<-walktrap.community(g)
V(g)$member<-member$membership
```

（3）关键用户

这里通过分析点的出度来研究哪些用户是转发的 10 个关键用户。代码如下：

```
degree.out<-degree(g,mode="out")
names(degree.out)<-V(g)$label
names(rev(sort(degree.out))[1:10])
[1] "yuxuejunliu2013"  "新闻哥"          "铁肩侠"          "邱永峥"
 [5] "盛大林"           "徐侠客"          "徐昕"            "大川"
 [9] "余以为"           "吉四六"
```

"yuxuejunliu2013" 是原始发布微博信息的用户名，其余的是转发次数最大的前 19 个用户。

3. 绘制关系网络图

把每个社群内的点统一为一种颜色，并作为 g 的 color 属性。代码如下：

```
member.color<-rainbow(length(unique(member$membership)),alpha=0.3)
V(g)$color<-member.color[member$membership]
```

其中 length(unique(member$membership) 是发现的社群数，member.color 是为每个社群分配的颜色信息，使用 member.color[member$membership] 把该颜色信息放于 g 的 color 属性中。

由于网络图中的点比较多，无法在一张图片中显示清楚，这里使用 svg 文件格式保存。svg 是网络通用的矢量图格式，其优点是在放大后图片不会失真，便于展示大规模的网络图。代码如下：

```
    svg(filename=paste(root,"demo.svg",sep=""),width = 14, height = 14)
    plot(g,vertex.size=1,layout=layout.fruchterman.reingold, vertex.
label=NA,vertex.color=V(g)$color)
    dev.off()
```

这里首先使用 svg 语句在根目录 root 下建立一个矢量图片文件 demo.svg，高和宽均为 14。然后使用 plot 函数绘制图形，其中 vertex.size 设置为 1 表示点大小为 1，layout=layout.

fruchterman.reingold 表示以 layout.fruchterman.reingold 方式布局网络图，vertex.label=NA 表示点不显示任何信息（默认显示点的 idx 号），vertex.color=V(g)$color 表示点的颜色设置为 V(g)$color。最后使用 dev.off() 语句关闭 demo.svg 文件，完成图像的输出。图 9-20 是微博传播网络图的截图。

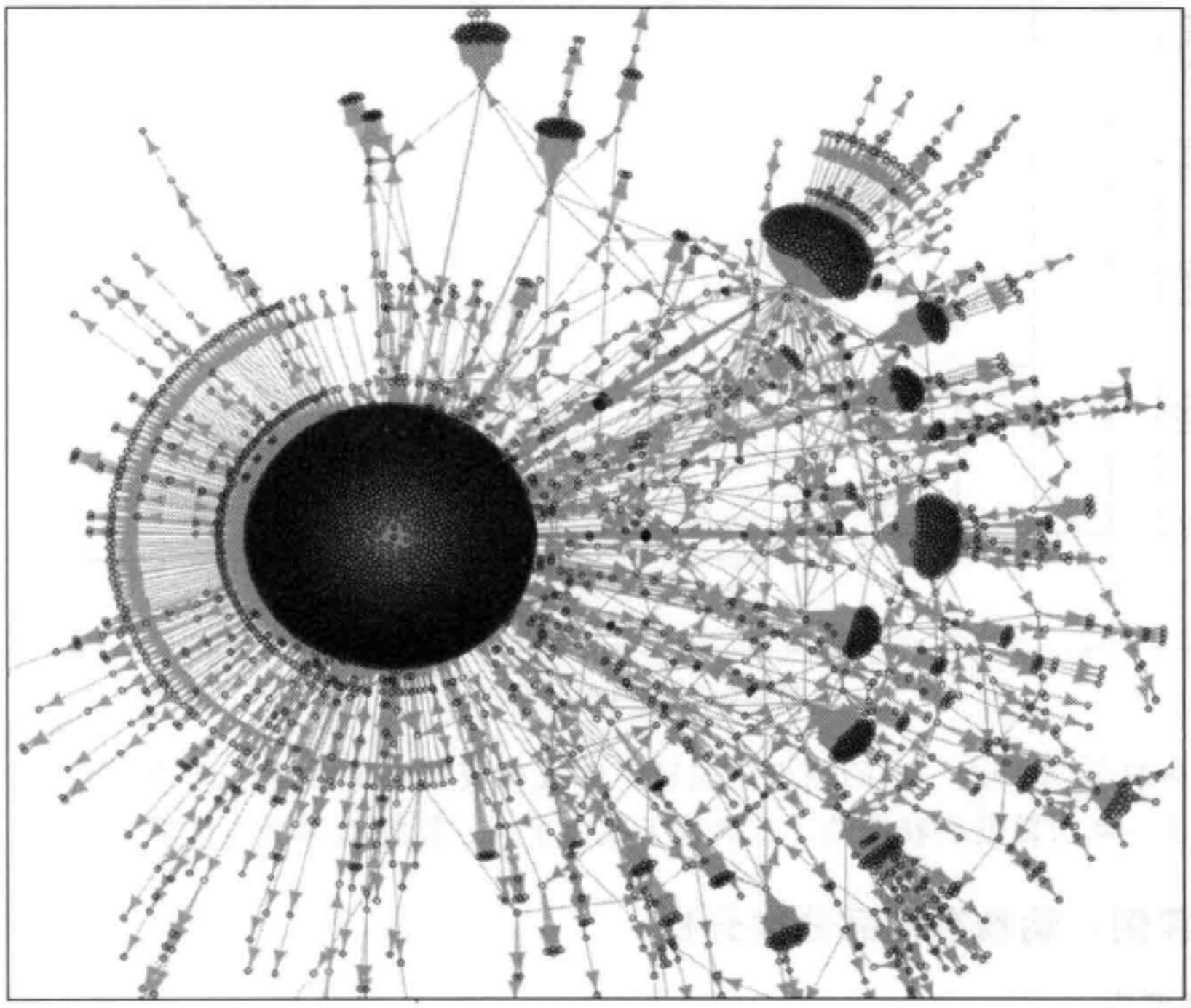


图 9-20　微博传播网络图

由截图可以看出，中间最大的红色圆形区域是社群 1，它代表发布该条微博的行为，可以看作是第一次传播。红色区域外有两圈明显的半圆圈（左侧），它表示微博的一次转发和二次转发行为。同时右侧有大量的社群被发现，应该是某些大 V 或活跃用户的转发。可以看出微博的传输不仅取决于发布微博用户的粉丝数，还要看在传播过程中被大 V 转发的情况，在被某些活跃用户转发后，其传播效果会更好。

4. 绘制微博转发统计图

在数据集中，发布微博的用户名为“yuxuejunliu2013”。由于微博是单向转发的，所以发布微博的用户与其他用户 i 的距离（注意 mode 是 out 形式，即使用出度的距离）就是用户 i 在接收到微博信息时，该微博被转发的次数。统计转发次数的代码如下：

```
orgid<-which(V(g)$label=="yuxuejunliu2013")# 原始发送微博的人的点 id
trans.num<-shortest.paths(g, v=orgid,mode="out")[1,]# 发布信息的人和其他人的距离，返回向量：转发次数
```

```
hist(trans.num,main=" 微博转发次数的频数分析 ",xlab=" 转发次数 ")
```

微博转发频次分析图如图 9-21 所示。

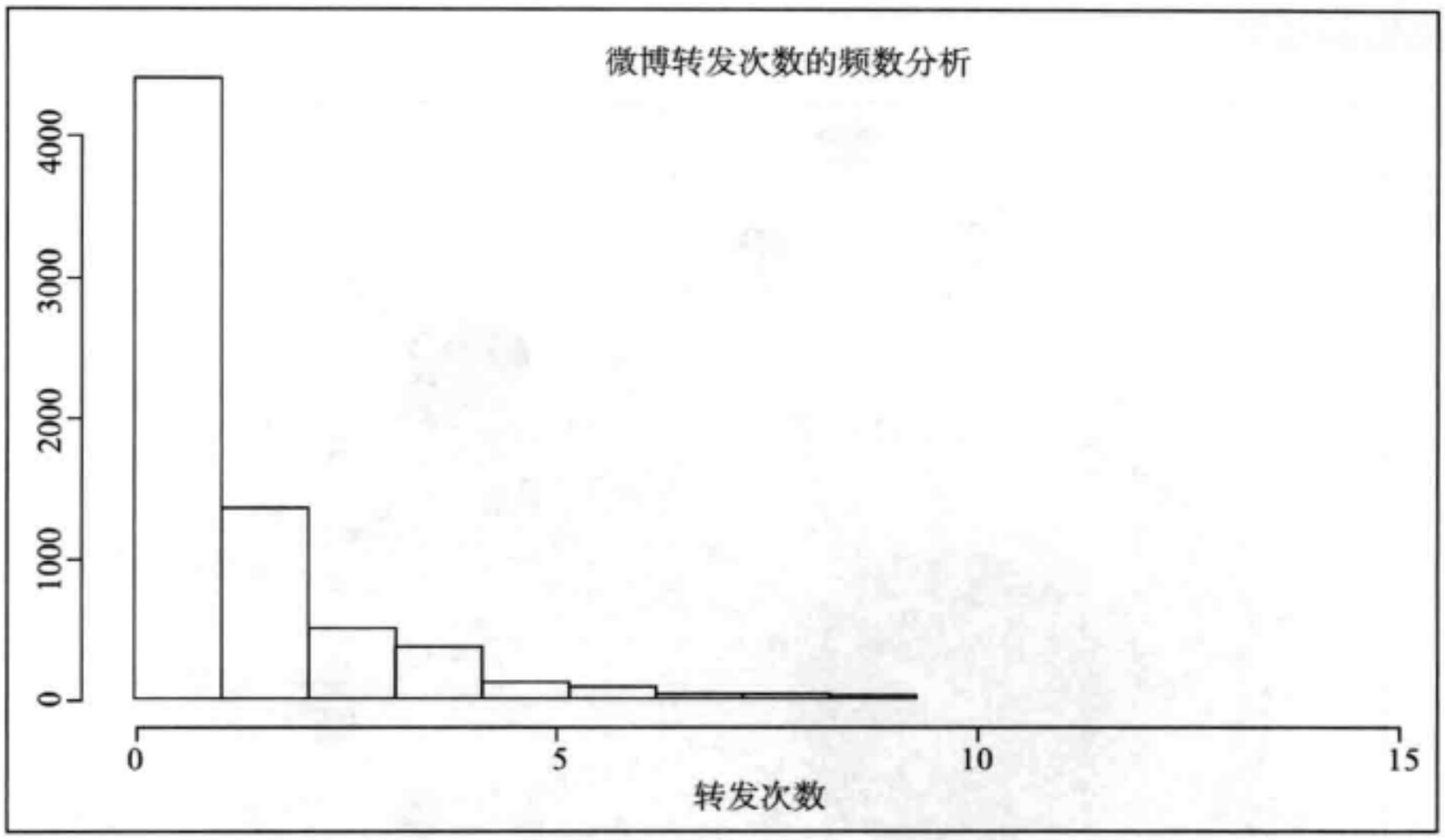


图 9-21 微博转发频次分析图

由图 9-21 可知，近一半的用户是通过一次发布获取该条微博信息的。还有小一半的用户是通过 2 ～ 4 次转发后获取的。在转发 5 次以上后，微博就很少再传播了。

9.3.6 实例：购物车商品分类分析

1. 数据集

某购物网站的购物车数据的源数据文件为“购物车数据 .csv”，使用 Excel 打开的截图如图 9-22 所示。

L22

	A	B	C
1		carid	item
2	1	162585	GDW_Tables.DW_CATEGORY_GROUPINGS
3	2	162765	DDM_RAM_T.CB_USER_BATCH_HOUR_USAGE
4	3	163665	GDW_Tables.DW_LSTG_ITEM_COLD
5	4	163455	DW_Monitor.DAILY_ACTV_VS_DELAY_QRY_CNT
6	5	162575	API_Tables.DW_API_MBL_APP
7	6	163555	DDM_RAM_T.CB_USER_BATCH_HOUR_USAGE
8	7	162595	GDW_Tables.DW_CAL_QTR
9	8	162695	GDW_Tables.DW_CAL_RTL_WEEK
10	9	162945	GDW_Tables.DW_LSTG_ITEM_COLD
11	10	163785	Working.DW_REGION_DEF_US_DT
12	11	162605	DDM_SEO_T.SEO_NS_CLICK_FACT
13	12	163345	GDW_Tables.GLBL_RPRT_BSNS_CTGRY_LKP
14	13	162635	API_Tables.DW_DEVELOPERS

图 9-22 数据集示例图

数据集的格式说明如表 9-28 所示。

表 9-28 数据集格式说明

列	列名称	说明
第一列	NA	行的序列号
第二列	carid	购物车 ID，相同购物车内的不同商品使用购物车 ID 来关联
第三列	item	商品名称，每行只包含一件商品，由“<商品分类名称>.<商品名称>”组成

1）使用 read.csv 函数读取源数据文件，并输入 init.data 对象中。代码如下：

```
root="D:/My Documents/写书/chap9/purchase/"
init.data<-read.csv(file=paste(root,"购物车数据.csv",sep=""),header =T);
```

2）依次轮询 init.data 的每一行数据，并使用 strsplit() 函数分割 item 中字符串的“.”，进而解析出商品名称和分类名称，并分别放于向量 item 和 category 中。代码如下：

```
item<-c();category<-c()
for(i in 1:nrow(init.data)){
    tmp<-strsplit(as.character(init.data$item[i]),split="\\.")[[1]]#使用.来分割
    item<-c(item,tmp[2])#商品名称
    category<-c(category,tmp[1])#商品品类
}
```

3）将字符串型向量 item 和 category 编号为 1 ～ n（itemid 和 categoryid），并存于 init.data 中。代码如下：

```
init.data$item<-item
init.data<-cbind(init.data,category=category)
init.data<-cbind(init.data,categoryid=as.numeric(as.factor(init.data$category)))
init.data<-cbind(init.data,itemid=as.numeric(as.factor(init.data$item)))
```

init.data 对象的最终结果如图 9-23 所示（前 10 行数据截图）。

	X	carid	item	category	categoryid	itemid
26	26	163405	CUST_SGMNTN	ESA01_Party	3	26
32	32	162595	CUST_SGMNTN	ESA01_Party	3	26
33	33	163155	CUST_SGMNTN	ESA01_Party	3	26
38	38	163125	CUST_SGMNTN	ESA01_Party	3	26
45	45	163425	CUST_SLR_SGMNTN_HIST	ESA01_Party	3	27
58	58	163635	CUST_SLR_SGMNTN_HIST	ESA01_Party	3	27
76	76	162675	CUST_SLR_SGMNTN_HIST	ESA01_Party	3	27
108	108	162835	CUST_SLR_SGMNTN_HIST	ESA01_Party	3	27
114	114	162635	CUST_SGMNTN	ESA01_Party	3	26
148	148	162655	CUST_SGMNTN	ESA01_Party	3	26

图 9-23 结果展示

2. 建立关系网络

以上数据并不能作为关系网络分析的输入数据，下面对其进行转化并建立关系网络。

（1）使用 cast 函数转化格式

通过 cast 函数得到购物车内的不同商品。代码如下：

```
library('reshape')
init.data<-cbind(init.data,value=1)
car.matrix<- cast(init.data,carid~itemid,value="value",fill=0)
car.matrix[,-1]<-ifelse(car.matrix[,-1]>=1,1,0)
```

最终的 car.matrix 格式如图 9-24 所示（以 car.matrix[93:101,1:20] 为例）。

	carid	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
93	163495	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0
94	163505	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
95	163515	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
96	163525	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
97	163535	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
98	163545	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
99	163555	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
100	163575	0	1	1	1	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0
101	163585	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

图 9-24 结果展示

其中，第一列为购物车 ID，列名称为 carid。其余各列为各商品是否在相应的购物车内，列名称为商品 ID init.data$itemid。所以每行的含义为每个购物车内所包含的商品。例如，init.data 的第 100 行表示为：编号为 163575 的购物车中包含商品 2、3、4、6、7。

（2）建立空关系网络并设置点数据

1）使用 graph.empty 函数建立空关系网络 init.data，并使用 add.vertices 函数添加相关的点信息，其中 num.item 是商品的数目。代码如下：

```
library(igraph)
num.item<-ncol(car.matrix)-1
data.g<-graph.empty(directed=F)
data.g<-add.vertices(data.g,num.item)
```

2）为点数据 V(data.g) 添加商品号以及商品分类号属性。代码如下：

```
category<-c();item<-c()
for(i in V(data.g)){
    category <- c(category,init.data$categoryid[which(init.data$itemid==i)[1]])
    item<-c(item,init.data$itemid[which(init.data$itemid==i)[1]])
}
V(data.g)$category <- category
V(data.g)$item <- item
```

（3）添加线数据

1）依次遍历每个购物车内的商品数据，读取每个购物车内包含的商品 ID，并存储于向量 item.i。

```
item.i<-as.numeric(colnames(car.matrix)[which(car.matrix[i,]==1)])
```

2）建立 item.i 内不同商品间的关联关系，并添加线信息。代码如下：

```
item.i.num<-length(item.i)
from<-c();to<-c()
```

```
    for(m in 1:(item.i.num-1)){
        from<-c(from,item.i[-c((item.i.num-m+1):item.i.num)])
        to<-c(to,item.i[-c(1:m)])
    }
    edges<-rbind(edges,matrix(c(from,to),nc=2))
```

3）添加线信息并设置线权重 E(data.g)$weight。代码如下：

```
    data.g<-add.edges(data.g,t(edges[-1,]))
    E(data.g)$weight <- count.multiple(data.g)
    data.g<-simplify(data.g, remove.multiple = TRUE, remove.loops = TRUE,edge.attr.
comb = 'mean')
```

最终的代码如下：

```
    for(i in 1:nrow(car.matrix)){# 遍历所有的 car 行
        item.i<-as.numeric(colnames(car.matrix)[which(car.matrix[i,]==1)])
        item.i.num<-length(item.i)
        from<-c();to<-c()
        for(m in 1:(item.i.num-1)){
            from<-c(from,item.i[-c((item.i.num-m+1):item.i.num)])
            to<-c(to,item.i[-c(1:m)])
        }
        edges<-rbind(edges,matrix(c(from,to),nc=2))
    }
    data.g<-add.edges(data.g,t(edges[-1,]))
    E(data.g)$weight <- count.multiple(data.g)
    data.g<-simplify(data.g, remove.multiple = TRUE, remove.loops = TRUE,edge.attr.
comb = 'mean')
```

3. 社群发现并绘制关系图

这里使用自旋转玻璃方法对其进行社群发现。代码如下：

```
member<-spinglass.community(data.g,weights = E(data.g)$weight)
V(data.g)$member<-member$membership
member.num<-length(table(V(data.g)$member)); member.num
[1] 4
```

由 member.num 的结果可知，数据被分为 4 个社群。然后依据商品分类 ID 为每个点分配一种颜色，使相同种类商品的点颜色相同。代码如下：

```
mem.col<-rainbow(length(unique(V(data.g)$category)),alpha=0.3)
V(data.g)$color<-mem.col[V(data.g)$category]
```

之后建立用于设置绘图分组（plot 函数的 mark.groups 参数）的列表对象 member.list。如果 10 个点中点 1 ～ 5 为一个社群，点 6 ～ 10 为一个社群，则 member.list 应设置为：

```
member.list=list(c(1:5),c(6:10))
```

然而在实际实例中，样本点被分为了 4 组，每组的点 idx 为 which(V(data.g)$member==i)，其中 i 的取值为 1 ～ 4。所以最后建立 member.list 的代码为：

```
member.list<-list()
for(i in 1:member.num){
    member.list<-c(member.list,list(which(V(data.g)$member==i)))
}
```

下面绘制具体的网络图，代码如下：

```
svg(filename=paste(root,"demo1.svg",sep=""),width = 14, height = 14)
plot(data.g,vertex.size=4,layout=layout.fruchterman.reingold,vertex.color=V(data.
g)$color,edge.width=scale(E(data.g)$weight,center=F)+1,mark.groups=member.list)
legend("topleft",legend=use.category.name,pch=16,col=mem.col,bty="n",cex=1)
dev.off()
```

通过 plot 函数的 mark.groups 参数设置点的分组，从而在绘图结果中显示不同颜色的区域。最终的结果如图 9-25 所示。

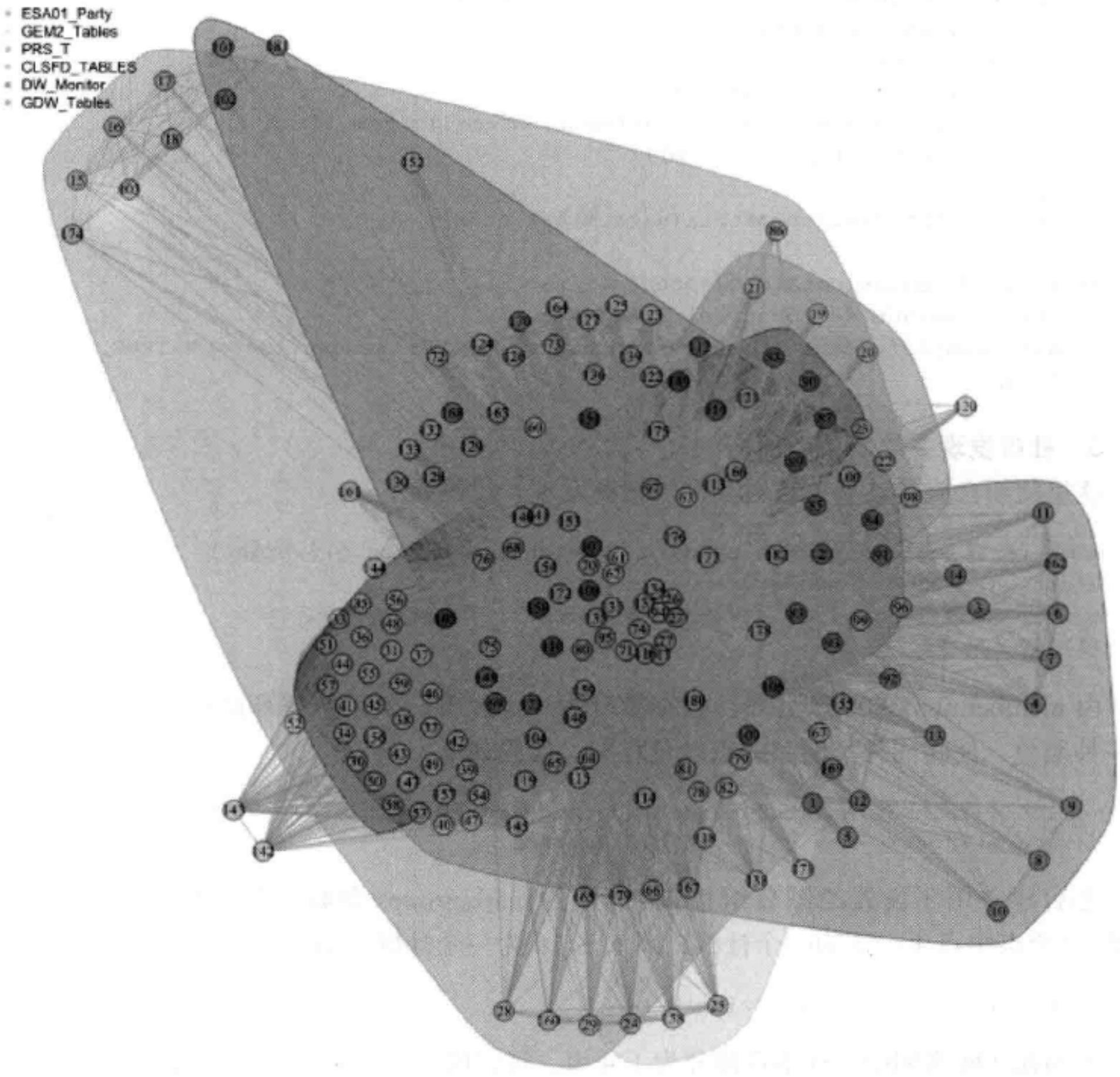


图 9-25 结果展示

在图 9-25 中，点的不同颜色则表示不同的商品分类，不同区域的底色代表不同的社群分组。

4. 绘制不同社群内的关系图

为了更清晰地观察分析结果，把不同社群内的点分别绘制成关系网络图。代码如下：

```
svg(filename=paste(root,"demo.svg",sep=""),width = 14, height = 14)
par(mfcol=c(1,4))
for(i in 1:length(table(member$membership))){
    tmp.g<-induced.subgraph(data.g,which(V(data.g)$member==i));V(tmp.g)
    member.list<-list()
    tmp.category<-as.numeric(names(table(V(tmp.g)$category)))
    for(j in tmp.category){
        member.list<-c(member.list,list(which(V(tmp.g)$category==j)))
    }
    plot(tmp.g,vertex.size=10,layout=layout.fruchterman.reingold,
edge.width=scale(E(tmp.g)$weight,center=F)+1,mark.groups=member.list,
vertex.label=V(tmp.g)$item)
    legend("topleft",legend=use.category.name,pch=16,col=mem.col,bty="n",cex=1)
}
dev.off()
```

绘制结果如图 9-26 所示。

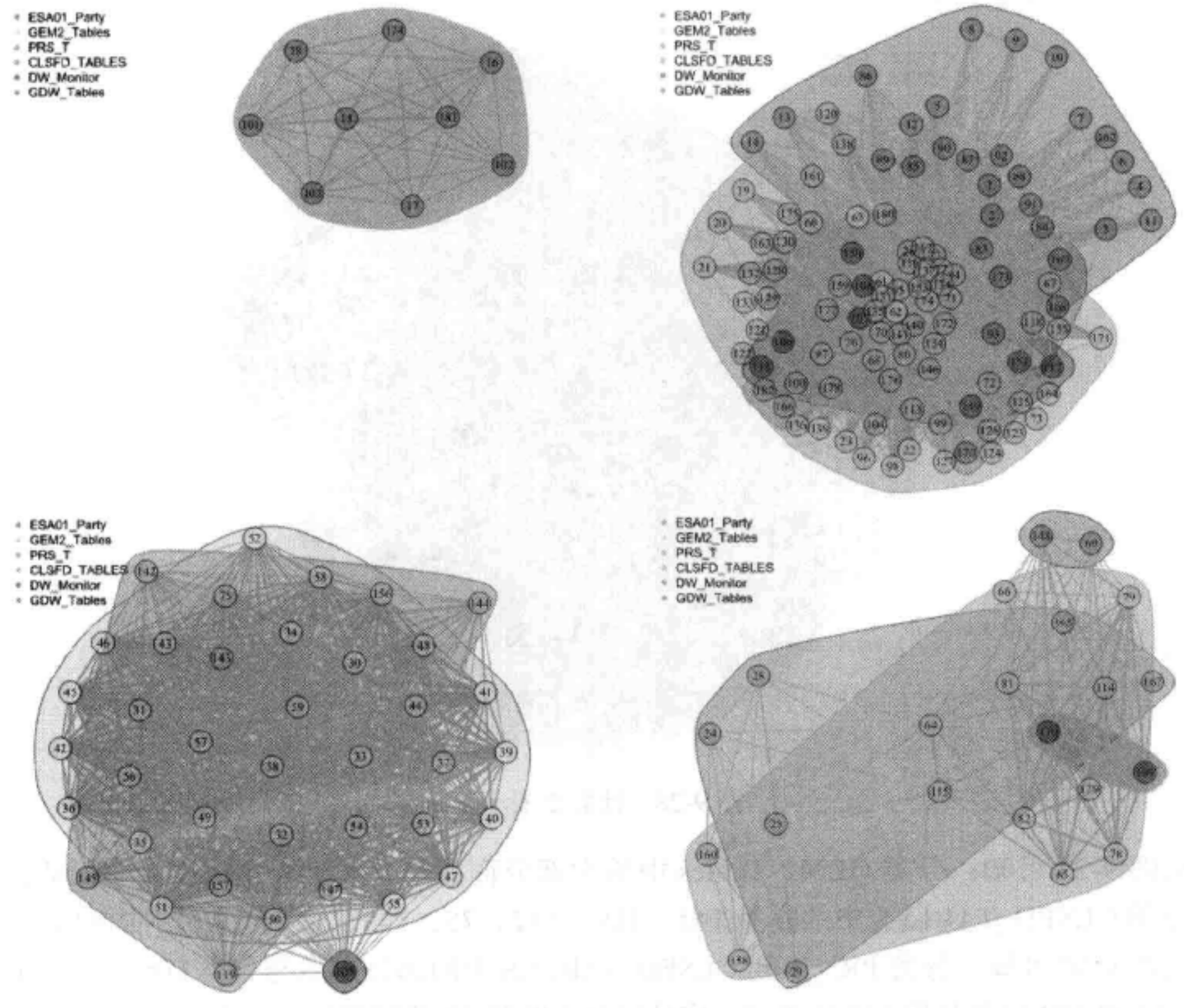


图 9-26　结果展示

这里共绘制 4 个子图，分别表示 1 ～ 4 个社群的关系图。同样，点的不同颜色仍代表不同的商品种类，同时子图内不同区域的底色也代表了不同的商品种类。

由图 9-27 可知，社群 1 就是商品分类 GDW_Tables 的大部分商品，该部分商品与其他分类商品的关联较少。

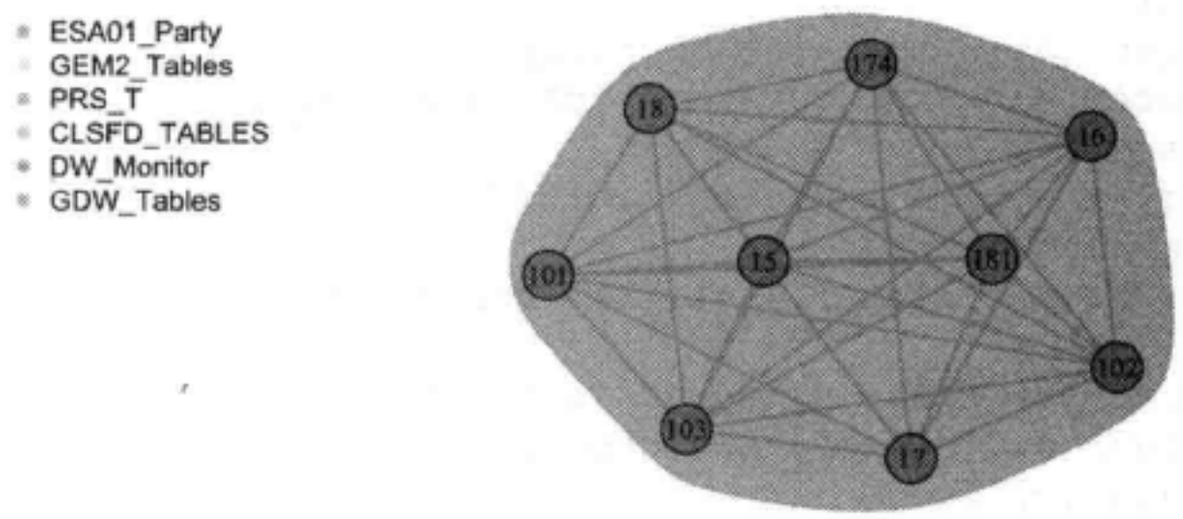


图 9-27 社群 1 展示图

由图 9-28 可知，分类 ESA01_Party 和 CLSFD_TABLES 中的部分商品关联较近，用户经常会同时购买这两组里的商品。

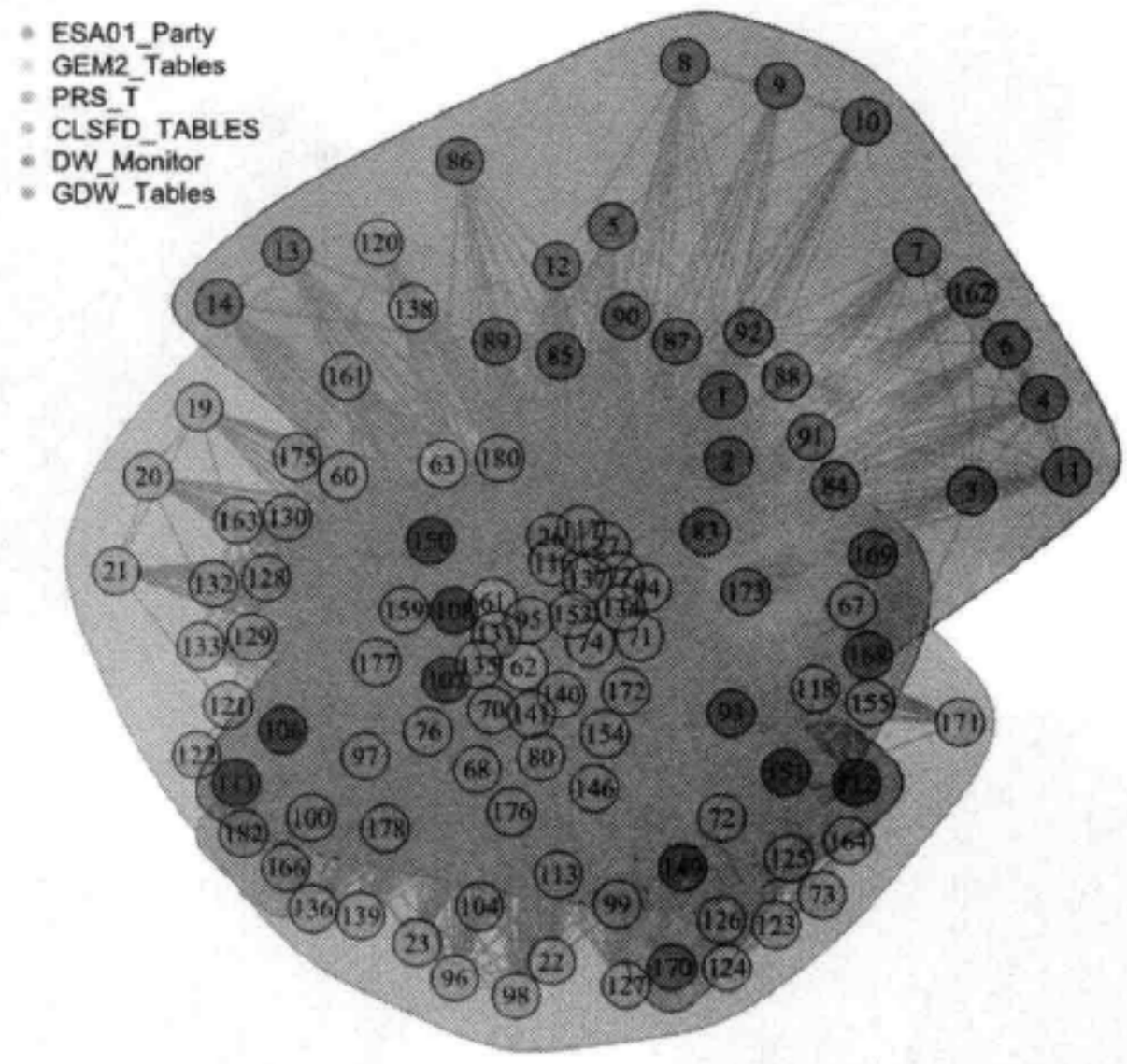


图 9-28 社群 2 展示图

由图 9-29 可知：分类 GEM2_Tables 中的大部分商品与分类 DW_Monitor 中的商品 105 以及分类 CLSFD_TABLES 中的商品 145、119、142、75、143、144 经常被一同购买。

由图 9-30 可知，分类 PRS_T 和 CLSFD_TABLES 中的部分商品与分类 DW_Monitor 中的商品 110 和 109 以及分类 GDW_Tables 中的商品 148 和 49 关联较近，用户经常会一起购买。

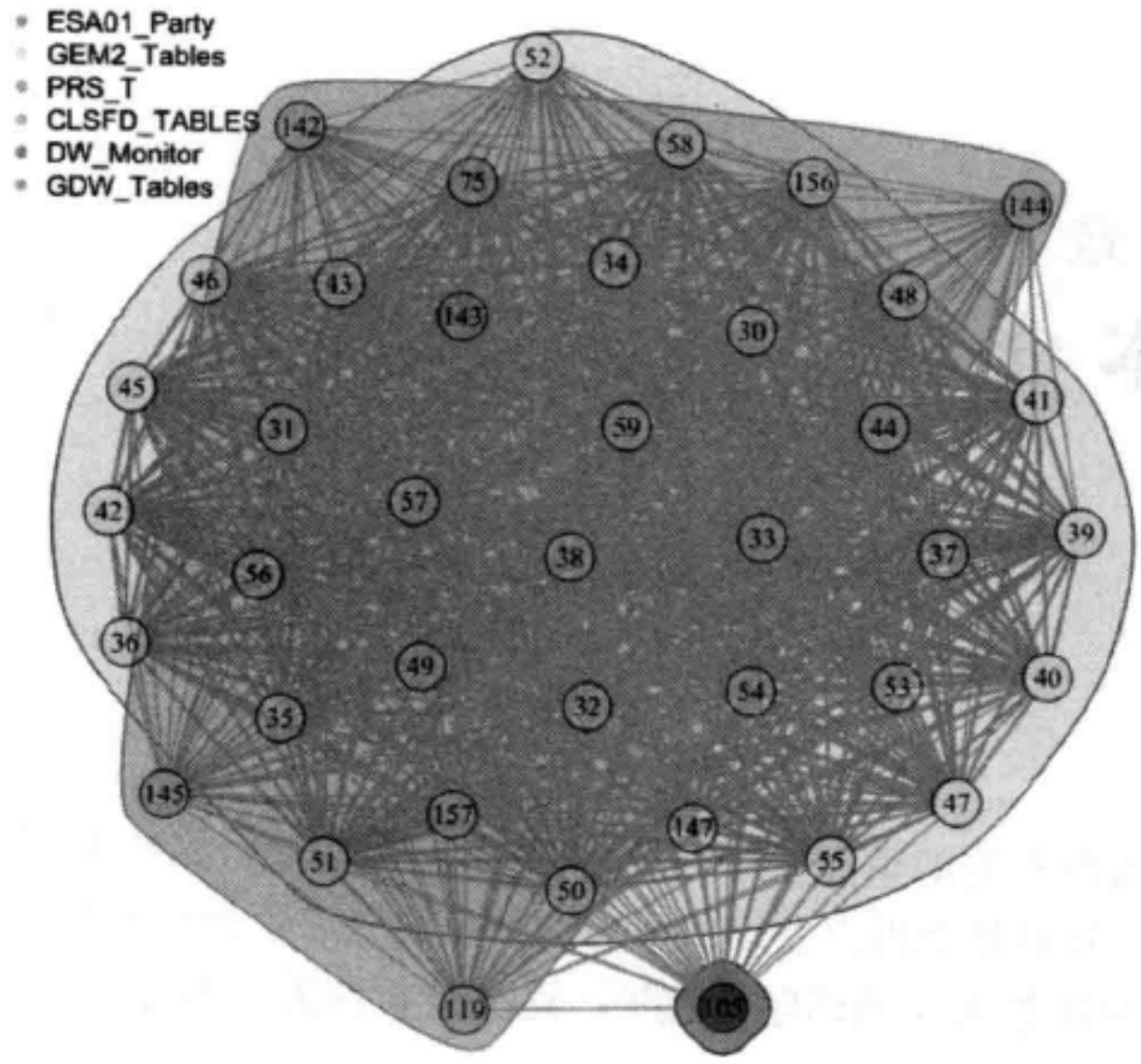


图 9-29　社群 3 展示图

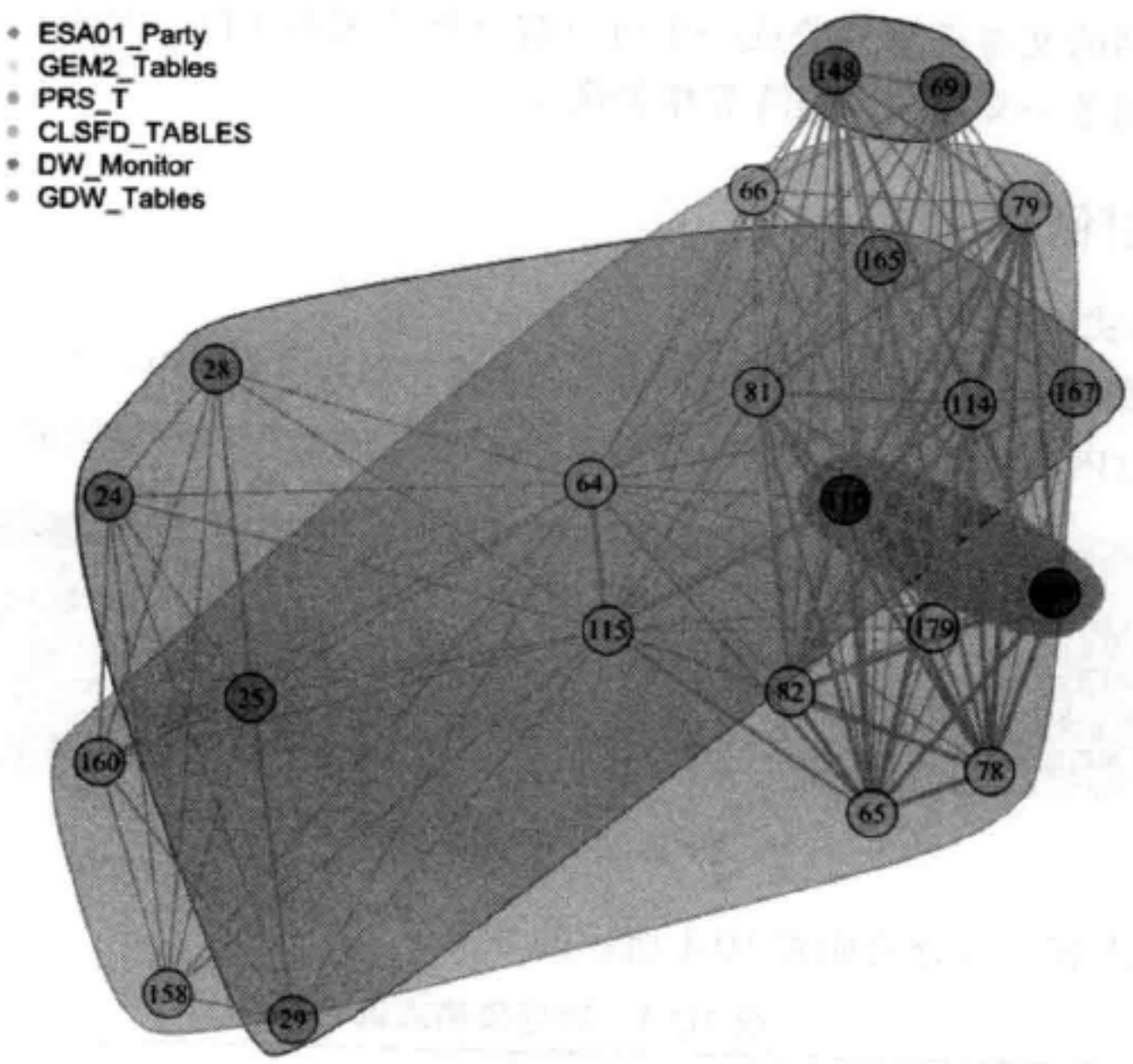


图 9-30　社群 4 展示图

第 10 章

文本分析

在互联网上，用户产生的信息主要是文字形式，通过分析这些文字内容可以获得很多重要的信息。例如，分析某个用户发出的所有微博信息（加上分词并统计各词汇出现的频率等），从而找出该用户比较关注的话题信息等。这类分析称为文本分析。

10.1 数据处理

本节把不规则的文本信息抽象成一个可以被分析的关系矩阵，并依次引入分词、词条、文档、语料库、词条—文档关系矩阵等相关概念。

10.1.1 数据引例

1. 数据集格式说明

为了更好地理解后续章节中提到的文本分析相关概念，这里先给出一个引例。一个用户发出/转发的微博的所有内容（共 7979 条微博信息），使用文本查看器打开后的截图如图 10-1 所示。

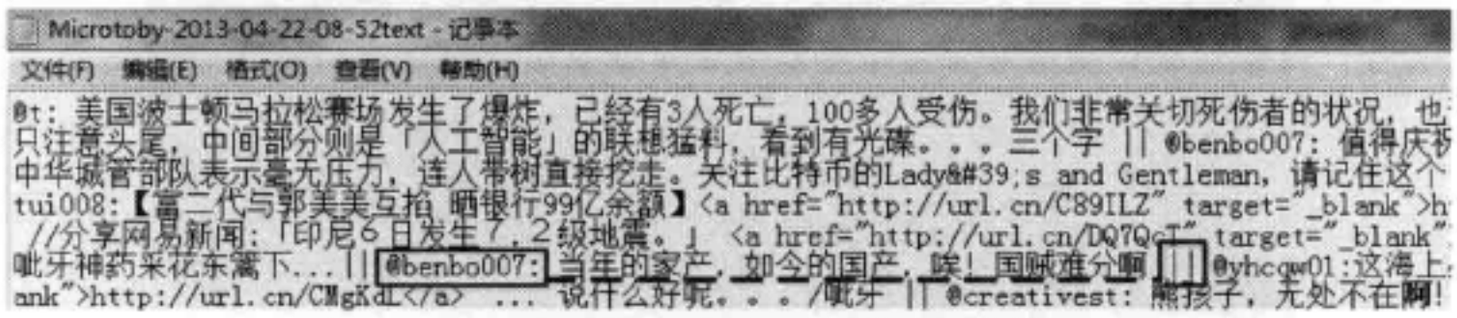


图 10-1 数据集说明

其中的文本内容可以分为如表 10-1 所示的 5 部分。

表 10-1 数据集格式说明

名称	说明	内容 / 格式	实例
开始标识符	标识该条微博的起始位置	@	@
用户名	用于说明该条微博发送给哪个用户	< 用户名 >	benbo007

（续）

名称	说明	内容 / 格式	实例
用户名与内容分隔符	用于区分用户名和微博内容	:	:
微博内容	微博的内容	< 微博内容 >	当年的家产，如今的国产，唉！国贼难分啊
结束标识符	标识出该条微博终止的位置	\|\|	\|\|

2. 数据读取

假设上面的原数据存储于“ D:\work\ 写书 \chap10\”目录下的 Microtoby-2013-04-22-08-52text.txt 文件中，下面使用 scan 函数读取相关文件内容：

```
root<-"D:/work/ 写书 /chap10/"
weibo<-scan(paste(root,"Microtoby-2013-04-22-08-52text.txt",sep=""),what="",sep
="@",encoding="UTF-8")
```

scan 函数用于读取文本文件中的所有数据，并把 @ 作为分隔符。得到的 weibo 对象是一个字符串型向量，向量内的每个元素 weibo[i] 是两个 @ 之间的内容。例如，weibo[1:2] 的展示如图 10-2 所示。

```
[1] "alexzxt:「技术的作用从短期来看往往被高估，但是从长期来看又往往容易被低估」 转的，认为有道理最后一次 [思考][思考] || "
[2] "kaifulee: [思考]阿兰刚离就开始看美女了 || "
```

图 10-2 读取后的结果展示

3. 解析用户名和微博内容信息

依据分隔符“:”解析出微博内容信息，以 weibo[1] 为例来说明，具体的思路如图 10-3 所示。

```
weibo[1]
 [1] "alexzxt:「技术的作用从短期来看往往被高估，但是从长期来看又往往容易被低估」 转的，认为有道理最后一次 [思考][思考] || "
strsplit(weibo[1],":")#strsplit函数依据分隔符：切分为列表对象
 [[1]]
 [1] "alexzxt"
 [2] "「技术的作用从短期来看往往被高估，但是从长期来看又往往容易被低估」 转的，认为有道理最后一次 [思考][思考] || "
unlist(strsplit(weibo[1],":"))#unlist函数把列表对象转化为向量对象
 [1] "alexzxt"
 [2] "「技术的作用从短期来看往往被高估，但是从长期来看又往往容易被低估」 转的，认为有道理最后一次 [思考][思考] || "
weibo.name<-unlist(strsplit(weibo[1],":"))[1]#读取用户名称
 [1] "alexzxt"
weibo[1]<-sub(weibo.name,"",weibo[1])#使用sub函数从weibo[1]中分离出微博内容
 [1] ":「技术的作用从短期来看往往被高估，但是从长期来看又往往容易被低估」 转的，认为有道理最后一次 [思考][思考] || "
```

图 10-3 解析用户名和微博内容信息示例

遍历 weibo 向量内的每个元素，最终的代码如下：

```
for(i in 1:length(weibo)){
    weibo.name<-unlist(strsplit(weibo[i],":"))[1]
    weibo[i]<-sub(weibo.name,"",weibo[i])
}
length(weibo)
[1] 7979
```

最后得到的 weibo 对象是只包含微博内容不再包含用户名的字符向量。它是以后进行文本分析的数据基础，共包括 7979 条微博信息。

10.1.2 分词和词条

在进行文本分析时，先要把一整段文字切割为具有最小语义的词条（trem）信息，该过程称为分词。例如，weibo[1] 的内容为：

```
":「技术的作用从短期来看往往被高估，但是从长期来看又往往容易被低估」转的，认为有道理最后一次 [思考][思考] ||"
```

分词后可以分为如下词条：

```
"技术" "的" "作用" "从" "短期" "来" "看" "往往" "被" "高估" "但是" "从" "长期" "来" "看" "又" "往往" "容易" "被" "低估" "转" "的" "认为" "有" "道理" "最后" "一次" "思考" "思考"
```

在 R 语言中，不同分词包的分词效果和效率均不同，这里介绍对中文分词较好的 Rwordseg 包，它使用 rJava 调用 Java 中文分词工具 Ansj 来实现分词。Ansj 是基于中科院 ictclas 中文分词算法的开源工具，它采用隐马尔科夫模型。由于该包要依赖 rJava 包和 Java 环境，所以在安装前需要安装好 JRE。R 软件一般需要使用 2.15.0 版本，至于其他高版本或低版本，笔者不能保证程序的稳定性。实现对 weibo 对象分词的代码如下：

```
library (rJava)
library (Rwordseg)
weibo.rwordseg<-segmentCN(weibo, nature = TRUE)
```

在加载 rJava 包和 Rwordseg 包后，就可以使用 segmentCN 函数进行分词操作。参数 nature 设置为 TRUE，表示输出对象包括词性的标识。例如，weibo.rwordseg [[1]] 的结果如图 10-4 所示。

n	uj	n	p	b	v	v	d	p	v	c	p	d	v	v
"技术"	"的"	"作用"	"从"	"短期"	"来"	"看"	"往往"	"被"	"高估"	"但是"	"从"	"长期"	"来"	"看"
d	d	a	p	v	v	uj	v	v	n	f	m	v	v	
"又"	"往往"	"容易"	"被"	"低估"	"转"	"的"	"认为"	"有"	"道理"	"最后"	"一次"	"思考"	"思考"	

图 10-4 segmentCN 函数的分词结果

其中 weibo.rwordseg 是一个包括 7979 个向量的列表对象。weibo.rwordseg [[i]] 是其中的第 *i* 个向量，表示用户发布的第 *i* 条微博的分词结果，其向量名称就是词性标识。例如，“技术”的名称是“n”(表示名词)，“来”的名称是“v”(表示动词)，“往往”的名称是“d”(表示形容词)。

在分词结果 weibo.rwordseg 中提取出其中的名词词条，代码如下：

```
weibo.rwordseg.n<-weibo.rwordseg
for(i in 1:length(weibo.rwordseg)){
      weibo.rwordseg.n[[i]]<-weibo.rwordseg[[i]][which(names(weibo.
rwordseg[[i]])=="n")]
}
```

其中的核心部分是在每个微博内容的分词结果向量中筛选出名字为 n 的元素。代码如下：

```
weibo.rwordseg[[i]][which(names(weibo.rwordseg[[i]])=="n")]
```

例如，第一条微博在提取名词词条后，weibo.rwordseg.n[[1]] 的结果如图 10-5 所示。

```
     n     n     n
"技术" "作用" "道理"
```

图 10-5 提取名词后的结果示例

10.1.3 语料库和文档

在 R 语言中，tm 包是文本分析的核心工具包，其中，语料库（corpus）是 tm 包处理所有文本的集合。例如，10.1.1 节数据引例中的 7979 条微博信息（分词后）就组成了一个语料库，每一条微博则可以视为一个文档（document）。

动态语料库（volatile corpus）会把语料库对象存储在内存空间中，在 tm 包中使用 Corpus 函数把原始资料源转化为动态资料库对象。静态语料库（permanent corpus）则把对象存储于外部数据库中。在 tm 包中，使用 PCorpus 函数和 filehash 包把动态语料库转化为静态资料库对象。

一个语料库中会包含多个文档。

1. Corpus 函数

Corpus 函数的基本参数如表 10-2 所示。

表 10-2 Corpus 函数的基本参数

参数	说明
x	原始资料源，可以使用一个向量或数据框，甚至是一个路径
readerControl	读取原始资料源的控制参数，这里必须声明 reader 和 language 两个内容，默认为 list(reader = x$DefaultReader, language = "en")。 reader 参数是指从资料源创立的文本文件，使用 getReaders() 可获取该参数的所有详细列表。例如，如果资料源是由多个 doc 文件组成的，则这里设置为 reader=readDOC()。 language 参数是读取时使用的字符库，常用的是 UTF-8

（1）原始资料源是一个字符向量

在 10.1.1 节数据引例中，被分词后的 7979 条微博（名词）信息存放于字符向量 weibo.rwordseg.n 对象中。首先使用 VectorSource 函数把该向量转化为可以被 Corpus 函数直接读取的数据，然后使用 Corpus 函数完成语料库的转化，并放于 ovid 对象中。代码如下：

```
ovid <- Corpus(VectorSource(weibo.rwordseg.n))
```

由于在使用 scan 函数读取原始文件时已经设置了编码，因此这里没有再设置 language 参数。注意，在向量 weibo.rwordseg.n 对象中，每个元素对应语料库中的一个文档。

（2）原始资料源是一个数据框

如果原始资料源是一个由 csv 文件读取的数据框对象，则先使用 DataframeSource 函数读入数据，然后使用 Corpus 函数完成语料库的转化。

（3）原始资料源是一个目录路径

如果原始资料源是“ D:\My Documents\ 写书 \chap10\data\”路径（windows 操作系统）

下的所有 doc 文件，如图 10-6 所示。

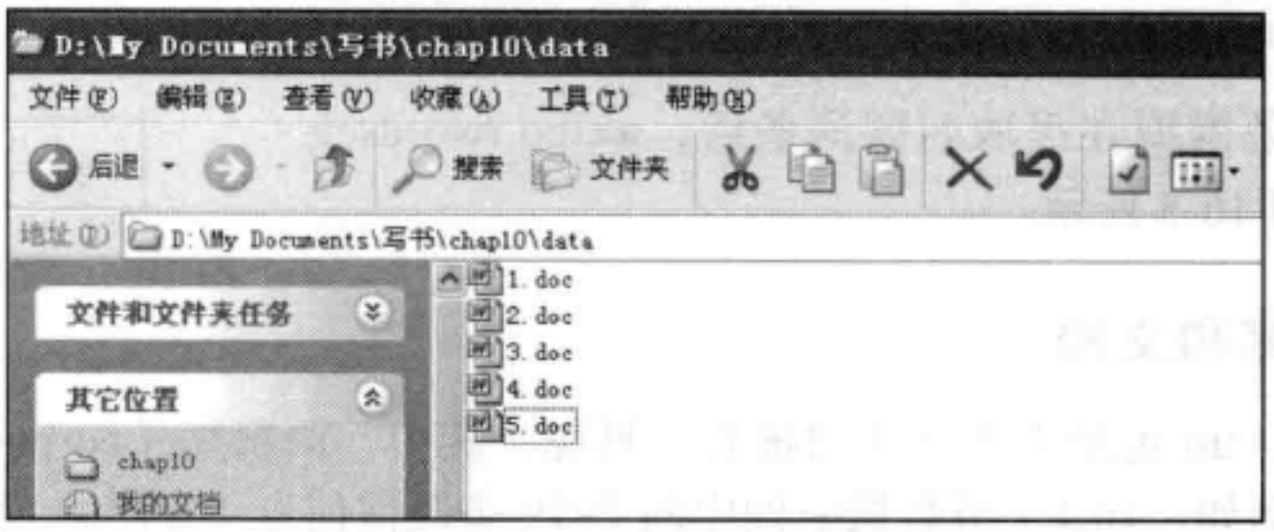


图 10-6 目录结构示意图

则首先使用 DirSource 函数设置该路径，然后设置 reader 为 readDOC()，最终代码为：

```
ovid <- Corpus(DirSource("D:/My Documents/写书/chap10/data/"),list(reader=
readDOC()))
```

注意，路径下的每个文件都对应语料库中的一个文档。

2. PCorpus 函数

由于静态语料库对象要存储于外部数据库中，所以 PCorpus 函数比 Corpus 函数多了一个设置参数，即设置数据库的参数 dbControl，它用来声明数据库资料源。其中 dbType 控制 filehash 包支持的数据库类型。数据库的支持可以有效地减少对内存的要求，但这样一来，数据的访问会受到硬盘读写能力的限制。

3. 语料库的提取和查看

函数 print() 和 summary() 可以概述出语料库的部分信息，而函数 inspect() 可以把所有的语料信息全部显示出来。例如，上文中的语料库 ovid 对象包含了 7979 条微博中的所有名称信息，使用代码 summary(ovid) 进行查看时，得到如图 10-7 所示的结果截图。

```
A corpus with 7979 text documents

The metadata consists of 2 tag-value pairs and a data frame
Available tags are:
  create_date creator
Available variables in the data frame are:
  MetaID
```

图 10-7 结果输出图

其中第一条“A corpus with 7979 text documents”表示语料库中包含了 7979 个文档。下面使用 inspect 函数查看语料库中的第一个文档 inspect(ovid[1])，结果如图 10-8 所示。

4. 文档处理

在创建好语料库以后，常常还需要修改内部的文档，如去白、删除停止词等。这里使用的主要函数是 tm_map(< 语料库 >, FUN =< 转化函数 >,< 转化函数的参数 >)，该函数可通过

maps 方式将转化函数实施到语料库中的每一个文档上。

```
A corpus with 1 text document

The metadata consists of 2 tag-value pairs and a data frame
Available tags are:
  create_date creator
Available variables in the data frame are:
  MetaID

[[1]]
技术
作用
道理
```

图 10-8　结果输出图

（1）删除词条

删除词条时，先设置转化函数为 removeWords，并设置后续的预删除字符向量。例如，显示语料库中文档 18（发送 / 转发的第 18 条微博中的名称词条）的内容，代码如下：

```
inspect(ovid[18])
```

输出结果如图 10-9 所示。

```
A corpus with 1 text document

The metadata consists of 2 tag-value pairs and a data frame
Available tags are:
  create_date creator
Available variables in the data frame are:
  MetaID

[[1]]
心机
砖
时候
市场
结果
牙
```

图 10-9　文档 18 的输出结果

如果认为“时候”和“结果”是无用词条，则可以使用如下代码删除：

```
ovid<-tm_map(ovid, FUN =removeWords, c("时候","结果"))
inspect(ovid[18])
```

处理后，词条“时候”和“结果”被替换为了空白，结果如图 10-10 所示。

（2）去除多余空白

设置转化函数为 stripWhitespace 可以删除空白的词条，代码如下：

```
ovid<- tm_map(ovid, FUN =stripWhitespace)
```

5. 文档过滤

可以使用 tm_filter(< 语料库 >, FUN =< 筛选函数 >,< 筛选函数的参数 >) 来筛选出符合某些条件的文档。例如，要搜索所有的文档，并找出包含词条“互联网”或“电子商务”的文档，相关代码为：

```
A corpus with 1 text document

The metadata consists of 2 tag-value pairs and a data frame
Available tags are:
  create_date creator
Available variables in the data frame are:
  MetaID

[[1]]
心机
砖

市场

牙
```

图 10-10 文档 18 的输出结果

```
ovid.search<-tm_filter(ovid, FUN = searchFullText, c("互联网","电子商务"))
```

对于文档处理和过滤中 FUN 参数的其他设置内容，请参考 tm 包的相关帮助文档，这里就不一一说明了。

10.1.4 词条—文档关系矩阵

1. 概念

前面完成了对文本数据（语料库）的逐级分割：首先把原始文本数据分割为文档，然后把每个文档分割为不同的词条。试想一下，如果文档代表行，词条代表列，那么行和列就组成了一个矩阵 A。通常，称矩阵 A 为词条—文档关系矩阵（document term matrix）。矩阵内的元素 a_{ij} 则表示第 j 个词条是否在第 i 个文档中出现，为 1 表示出现，为 0 表示不出现。

在 tm 包中使用 DocumentTermMatrix(< 语料库 >) 实现语料库到词条—文档关系矩阵的转化。其中行名称是文档的序号，列名称是词条。下面的代码把 10.1.3 节引例中的语料库 ovid 转化为关系矩阵 dtm：

```
dtm <- DocumentTermMatrix(ovid)
```

图 10-11 是语料库—文档—词条—关系矩阵的示例图，经过一步一步处理，实际上已把一个不规则的文本信息转化为了可用于数据分析的矩阵信息。

在图 10-11 中，因为 92 号文档的内容是“这个加速度应该要超越人体承受极限了吧”，它包含词条“加速度”，所以对应的矩阵元素为 1。读取相关信息的 R 语言代码如下：

```
col.num<- which(colnames(dtm)=="加速度"); col.num
[1] 406
inspect(dtm[92,col.num])
```

结果如图 10-12 所示。

在关系矩阵 dtm 中，用 dtm$dimnames$Terms 或 colnames(dtm) 可以查看所有的词条信息。例如，想要找到 dtm 中词条“加速度”所在的列序号，可用如下代码：

```
col.num<- which(colnames(dtm)=="加速度")
```

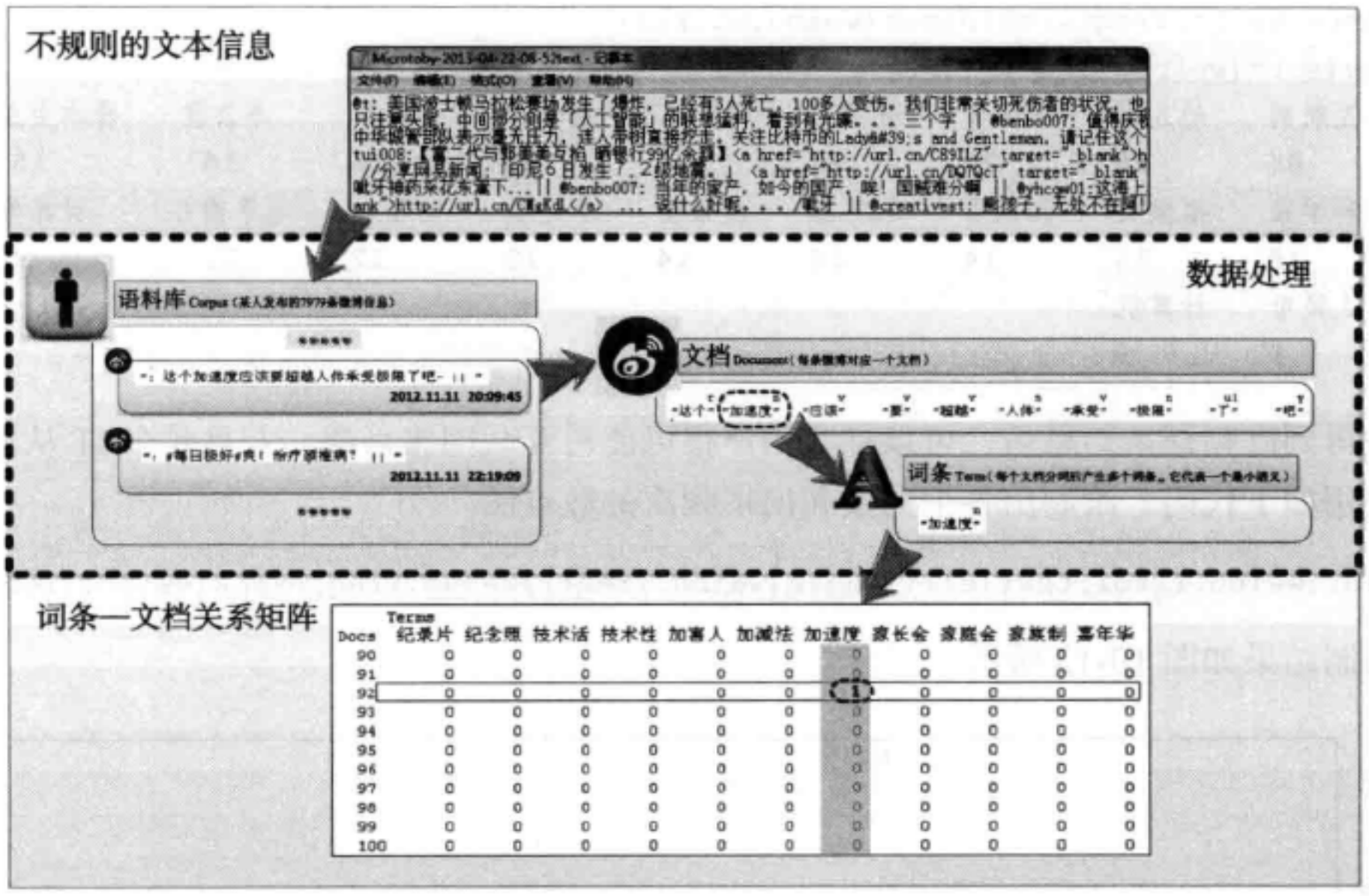


图 10-11 语料库—文档—词条—关系矩阵的关系示例图

```
A document-term matrix (1 documents, 1 terms)

Non-/sparse entries: 1/0
Sparsity           : 0%
Maximal term length: 3
Weighting          : term frequency (tf)

    Terms
Docs 加速度
  92      1
```

图 10-12 读取结果

2. 相关操作

上面的关系矩阵和 R 中的矩阵很类似，R 中的很多矩阵处理函数均可以应用到该类对象上。或者使用 as.matrix() 函数把关系矩阵直接转化为普通的矩阵。此外 tm 包还提供了一些额外的处理函数。

（1）统计包含词条的文档数

1）在统计包含词条的文档数时，首先要把关系矩阵 dtm 转接为普通矩阵 weibo.matrix：

```
weibo.matrix<-as.matrix(dtm)
```

2）使用 apply 函数统计各个词条所在的文档数：

```
weibo.freq<-apply(weibo.matrix,2,sum)
```

3）依据文档数排序，找到频次出现最多的 20 个词条：

```
names(weibo.freq)<-colnames(weibo.matrix)
rev(sort(weibo.freq))[1:20]
```

互联网	幼儿园	程序员	开发者	笔记本	消费者	全世界	办公室	资本主义
85	28	28	24	21	16	16	16	15
科学家	摄像头	浏览器	服务器	发布会	小时候	创始人	电子商务	世界杯
15	14	14	14	14	12	12	11	9
人民币	计算机							
9	9							

在得到高频词条结果后，可以推测用户很可能对互联网感兴趣，并且是个 IT 从业者。使用如下代码，绘制出各个词条和出现频次的散点图。

```
plot(weibo.freq);text(c(1:length(weibo.freq)),weibo.freq,names(weibo.freq))
```

绘制结果如图 10-13 所示。

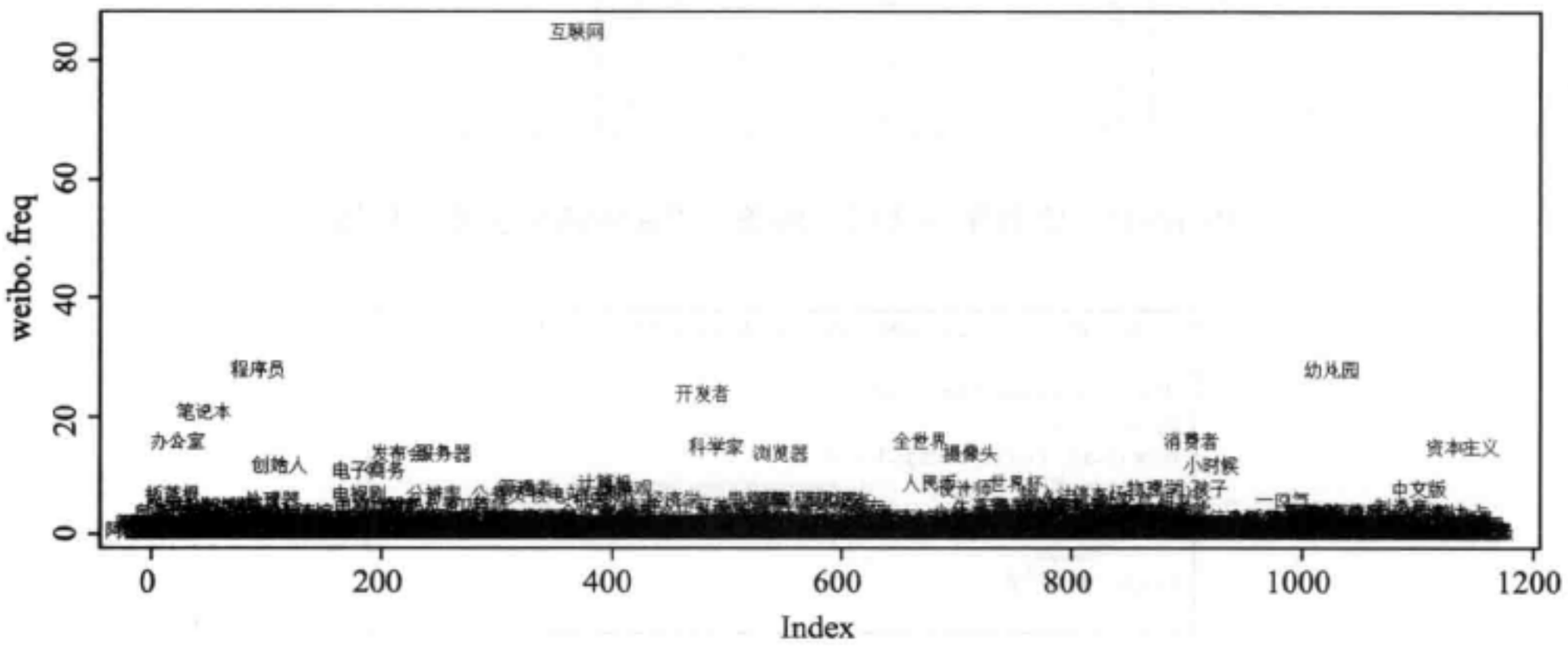


图 10-13 词条和出现频次的散点图

（2）查找高频次词条

使用 findFreqTerms() 函数可以直接提取在文档中出现频次较高的词条。例如，要找出在 10 个以上的文档中出现过的词条，可采用如下代码：

```
findFreqTerms(dtm,10)
[1] "办公室"  "笔记本"  "程序员"  "创始人"  "电子商务"  "发布会"  "服务器"  "互联网"
[9] "开发者"  "科学家"  "浏览器"  "全世界"  "摄像头"  "消费者"  "小时候"  "幼儿园"
[17] "资本主义"
```

可见该用户对如上话题比较感兴趣。

（3）删除稀疏词条

tm 包中的 removeSparseTerms 函数提供了一种删减稀疏条目的方法。例如，有些词条只在很少的文档中出现过，那么就可以将其删除。一般来说，这样做不会对矩阵的信息继承带

来显著的影响。一般认为：如果一个词条在语料库中所属文档的数目占整体不到 0.03%，则该词条为稀疏词条。在本节示例中 dtm 有 7979 个文档，如果词条出现的文档数小于等于 2（7979 × 0.0003），就称为稀疏词条。删除该类词条的代码如下：

```
dtm.rem.lowterm<-removeSparseTerms(dtm, (1-0.0003)); dtm.rem.lowterm
A document-term matrix (7979 documents, 144 terms)
Non-/sparse entries: 794/1148182
Sparsity           : 100%
Maximal term length: 4
Weighting          : term frequency (tf)
```

结果显示，原来 dtm 中有 1161 个词条，现在被删除至 144 个词条。

（4）寻找相关性词条

有些词条的相关性很大，如词条“互联网”和“因特网”的所属文档信息为向量 term1 和 term2，那么它们的相关性如何？可用如下代码分析：

```
term1<-weibo.matrix[,which(colnames(dtm)==" 互联网 ")]
term2<-weibo.matrix[,which(colnames(dtm)==" 因特网 ")]
cor(term1,term2)
[1] 0.1345911
```

结果显示，“互联网”和“因特网”两个词条的相似度约为 0.13。

在 tm 包中，使用 findAssocs(dtm，< 词条 >,n) 函数可以在关系矩阵 dtm 中找到与 < 词条 > 相关性大于 n 的其他词条信息。例如，要找到与“互联网”相关性大于 0.1 的词条，代码如下：

```
findAssocs(dtm," 互联网 ", 0.1)
  互联网     流量     消费者     搜索引擎     因特网     计算机
     1.00     0.19     0.14     0.13     0.13     0.11
```

可见“流量”、“消费者”、“搜索引擎”、“因特网”以及“计算机”与“互联网”同时出现的概率较大，即其相关性较大。

10.2 实例：QQ 群消息历史分析

10.2.1 数据集

这里使用的数据集是作者创立的“R 语言数据挖掘分析”QQ 交流群（238649264）中的部分消息记录，其格式如图 10-14 所示（使用文本查看器打开后的部分截图）。

文件的头部是 6 行数据说明信息，对于数据分析无任何作用。其余文本内容可以分为 3 部分，如表 10-3 所示。



图 10-14 QQ 交流群部分消息记录

表 10-3 数据集格式说明

名称	内容 / 格式	实例
发布时间	年 (4 位)– 月份（2 位）– 日期（2 位）小时（2 位）– 分（2 位）– 秒（2 位）	2012-06-11 13:31:57
发布人	发布时间后的信息	345530857(345530857)
留言内容	处于时间和发布人信息一行的下几行	BI 是商务智能的意思吧

使用如下代码读入上面的数据：

```
root="D:/work/ 写书 /chap10/qq.union/"
file=paste(root," R 语言数据挖掘分析 (238649264).txt",sep="")
    file.data<-scan(file,what="",sep="\n",encoding="UTF-8")
```

使用 scan 函数得到对象 file.data 的样式，如图 10-15 所示。

```
> file.data[1:12]
 [2] "=========================================================="
 [3] "消息分组:我的QQ群\n=========================================================="
 [4] "消息对象:BI商务智能"
 [5] "=========================================================="
 [6] "2012-06-11 13:31:57 345530857(345530857)"
 [7] "BI是商务智能的意思吧"
 [8] "2012-06-11 13:32:07 781385715(781385715)"
 [9] "在那？ \n"
[10] "2012-06-11 13:32:49 345530857(345530857)"
[11] "在北京 "
[12] "2012-06-11 13:32:58 345530857(345530857)"
```

图 10-15 读取的结果

使用规则表达式解析发布时间 time、发布人 user.name 和留言内容 text 信息，并存储于数据框 data 中。首先读取 file.data[6]，即文本源文件的第 6 行，解析其是否包含时间规则表达式。代码如下：

```
reg.time<-regexpr("[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]+:[0-9]+:[0-9]+",file.data[i])
```

其中“[0-9]{4}-[0-9]{2}-[0-9]{2}”表示 <4 位的年份信息 > - <2 位的月份信息 > - <2 位的日期信息 >。“[0-9]+:[0-9]+:[0-9]+”表示 <1 ～ 2 位的小时信息 >：<1 ～ 2 位的分信息 >：<1 ～ 2 位的秒信息 >。如果 reg.time 等于 1，则表示该行信息包含时间和发送人信息。使用

下面代码来读取时间信息：

```
begin<-reg.time;end<-reg.time+attr(reg.time,"match.length")-1
time<-substr(file.data[i],begin,end)# 读取时间信息
```

并使用如下代码把该行其余的信息解析为用户信息：

```
begin<-as.numeric(reg.time+attr(reg.time,"match.length")+1);end<-nchar(file.
data[i])
user.name<-substr(file.data[i],begin,end)# 读取用户名信息
```

如果 reg.time 不等于 1，则表示该行信息不会包含时间和发送人的信息。此时把该行设置为留言信息，最终遍历所有有效数据的代码如下：

```
data<-data.frame(uesr.name=c(),time=c(),text=c())
time<-c();user.name<-c();text<-c()
for(i in 6:length(file.data)){# 遍历有效的行数据
      reg.time<-regexpr("[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]+:[0-9]+:[0-9]+",file.
      data[i])
      if(reg.time==1){# 该行读取到了时间信息
           data<-rbind(data,data.frame(time=time,user.name=user.
           name,text=text))
           text<-c()
           begin<-reg.time;end<-reg.time+attr(reg.time,"match.length")-1
           time<-substr(file.data[i],begin,end)# 读取时间信息
           begin<-as.numeric(reg.time+attr(reg.time,"match.length")+1);end<-
           nchar(file.data[i])
           user.name<-substr(file.data[i],begin,end)# 读取用户名信息
      }else{
           text<-paste(text,file.data[i])
      }
}
data$text<-as.character(data$text)
data$user.name<-as.character(data$user.name)
```

最后得到的 data 数据框格式如图 10-16 所示。

```
> data[1:3,]
                 time             user.name                    text
1 2012-06-11 13:31:57 345530857(345530857)  BI是商务智能的意思吧
2 2012-06-11 13:32:07 781385715(781385715)              在那？\n
3 2012-06-11 13:32:49 345530857(345530857)              在北京
```

图 10-16 data 数据框格式

10.2.2 数据集格式转化

数据框 data 中的一行内容表示一条留言信息（包括留言时间、用户、内容），我们想使用文本分析研究其中的重要用户和重要话题，这里先把每个用户的所有留言合并，产生新的数据框信息 user.text。代码如下：

```
user.name<-unique(data$user.name)
text<-c();text.num<-c()
for(i in 1:length(user.name)){
      text.i<-data$text[which(data$user.name==user.name[i])]# 用户 i 的所有留言向量
```

```
        text.i.num<-length(text.i)# 用户 i 发几次留言
        for(j in 1:text.i.num){
              text[i]<-paste(text[i],text.i[j],sep=" ")# 把所有的留言向量合并成一个元素
        }
        text.num[i]<-text.i.num
}
user.text<-data.frame(user.name=user.name,text=text,text.num=text.num)
user.text$user.name<-as.character(user.text$user.name)
user.text$text<-as.character(user.text$text)
```

首先，读取所有的用户名 user.name<-unique(data$user.name)，并遍历它读取第 i 个用户的所有发言信息 text.i<-data$text[which(data$user.name==user.name[i])]，然后使用 text[i]<-paste(text[i],text.i[j],sep=" ") 把每个留言 j 合并为字符串 text[i]，再使用 text.num[i] 记录该用户的留言次数，最后得到的 user.text 格式如图 10-17 所示。

```
> user.text[1:2,]
                user.name                                                                  text text.num
1 345530857(345530857)   BI是商务智能的意思吧 在北京 工作地点在长沙 就是不知道啊 在问你们啊        5
2 781385715(781385715)                        NA 在那？\n 多少钱一个月啊\n [表情] 我也不知道哦        4
```

图 10-17 数据集格式

其中，user.name 列表示发言用户名。text 列是由该用户所有留言组成的字符串，每条留言间使用空格分离。text.num 列是该用户的留言次数。例如第一行表示用户名为“345530857(345530857)”的用户，留言次数为 5，留言内容依次是：“BI 是商务智能的意思吧”、“在北京”……“在问你们啊”。

10.2.3 留言文本的分词操作

加载分词时使用的 rJava 和 Rwordseg 包，会把每个用户的所有留言数据 user.text$text 向量进行分词，从而得到对象 rwordseg。代码如下：

```
library (rJava)
library (Rwordseg)
rwordseg<-segmentCN(user.text$text,nature = TRUE)
```

在包括所有词性的分词数据 rwordseg 中提取出名词 rwordseg.n。代码如下：

```
rwordseg.n<-rwordseg
for(i in 1:length(rwordseg)){
      rwordseg.n[[i]]<-rwordseg[[i]][which(names(rwordseg[[i]])=="n")]
}
```

每个用户 user.text$user.name[i] 的所有留言中的名词分词信息为向量 rwordseg.n[[i]]。例如，用户 user.text$user.name[1] 的所有留言信息如图 10-18 所示。

```
> user.text$text[1]
[1] "  BI是商务智能的意思吧  在北京  工作地点在长沙  就是不知道啊  在问你们啊 "
```

图 10-18 某条留言信息

分词后的信息如图 10-19 所示。

```
> rwordseg.n[[1]]
     n      n      n      n
"商务" "智能" "意思" "地点"
```

图 10-19 分词后的结果

10.2.4 分析常用话题词汇

分析常用话题词汇时，和上一节类似，首先加载 tm 包并使用 Corpus 函数建立语料库 ovid。由于“图片”和“表情”两个词汇是用户上传图片和发送表情时系统自动记录的词条，并无实际意义，因此这里使用 tm_map 函数删除词条。代码如下：

```
library(tm)
ovid <- Corpus(VectorSource(rwordseg.n))
ovid<- tm_map(ovid, FUN =removeWords,c("图片","表情"))# 删除 qq 系统词
```

用 DocumentTermMatrix 函数把语料库 ovid 转化为文档—词条关系矩阵 dtm。代码如下：

```
dtm <- DocumentTermMatrix(ovid)# 创建词条 - 文档关系矩阵
```

然后统计每个词条所在的文档数（实际就是有多少用户的留言中包括了该词条），并提取前 10 个典型的名词词条。代码如下：

```
qq.matrix<-as.matrix(dtm)
qq.freq<-apply(qq.matrix,2,sum)
qq.freq.top<-rev(sort(qq.freq))[1:30]
```

最后直接绘制出所有词条—所在文档数散点图。代码如下：

```
plot(qq.freq);text(c(1:length(qq.freq)),qq.freq,names(qq.freq))
```

绘制结果如图 10-20 所示。

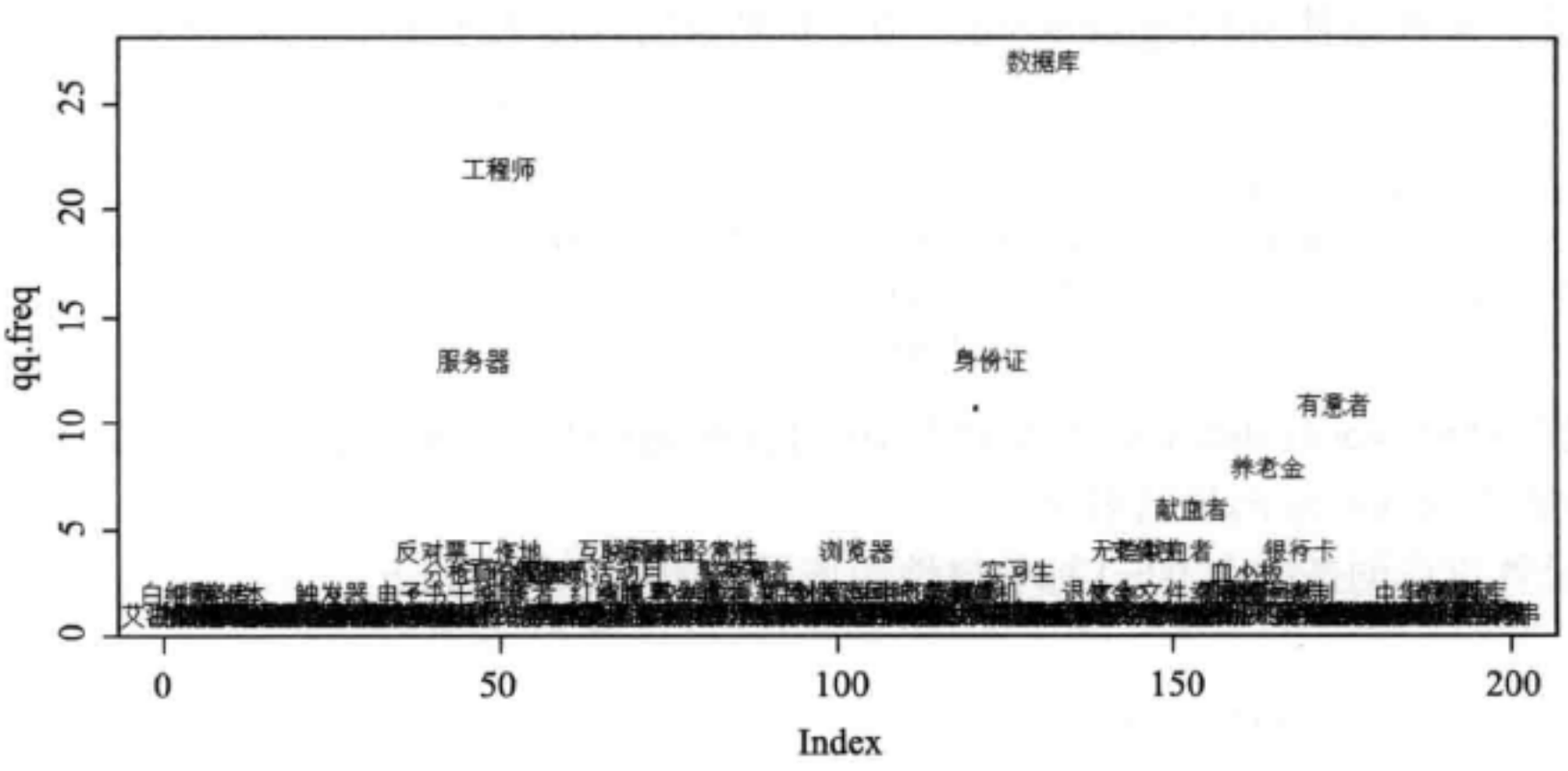


图 10-20 词条—所在文档数散点图

10.2.5 建立用户－词条间的网络关系图

建立用户和该用户发表的名词词条间的有向网络关系图 g.dir，首先把用户名 username 设置为 from，名词分词 rwordseg.n 设置为 to。

例如，如果 user.text$user.name[1]=“011110”,rwordseg.n[[1]]=c(“汽车","晚上","互联网");user.text$user.name[2]=“林(0900921)”,rwordseg.n[[1]]=c(“手机","学生")。那么有向网络关系图中的 from 和 to 可以分别设置为 from=c(“011110”,“011110”,“011110”,“林(0900921)”,“林(0900921)”)，to=c(“汽车","晚上","互联网","手机","学生")。

上述过程的实现代码如下：

```
from<-c();to<-c()
for(i in 1:length(user.text$user.name)){
      from<-c(from,rep(user.text$user.name[i],length(rwordseg.n[[i]])))
      to<-c(to,rwordseg.n[[i]])
}
from[which(from=="")]<-"数据集内无用户名"
```

如果解析出的用户名为空，则填充为默认值"数据集内无用户名"。最后使用第 9 章中的 init.igraph 函数建立网络关系图。代码如下：

```
library(igraph)
g.dir<-init.igraph(data.frame(from=from,to=to),T)
```

10.2.6 绘制重点词条和用户的网络图

1. 绘制重点词条网络图

这里以点的入度为度量筛选出重点词条信息，入度越大，表示该词条被越多的人发布，即该话题带动了更多的用户参与讨论。

1）设置筛选条件 std.degree.words=10，如果点的入度大于 10，则列为重点词条。代码如下：

```
std.degree.words=10;
words.index<-(degree(g.dir,mode="in")>=std.degree.words)
words<-degree(g.dir,mode="in")[words.index]
names(words)<-V(g.dir)[words.index]$label
```

这里的对象 words.index 是入度大于 10 的点的 idx 号，words 是重点词条的入度，然后把词条内容作为 words 向量的名字。

2）设置重点词条的点 label 标签属性为该词条内容。代码如下：

```
label=NA
label[words.index]<-names(words)
```

非重点词条的 size 属性（点大小）为 1，重点词条为 2 ～ 4：

```
V(g.dir)$size=1# 一般词汇和用户的大小为 1
```

```
max.d<-max(words)
min.d<-min(words)
V(g.dir)[words.index]$size=2*(words-min.d)/(max.d-min.d)+2# 关键点词汇的大小为 2-4
```

非重点词条的 color 属性（点颜色）为白色，重点词条为红色：

```
V(g.dir)$color="white"
V(g.dir)[words.index]$color="red"
```

3）在 svg 文件中输出关系图，注意，svg 要显示中文必须设置 vertex.label.family 参数为 "GB1"。代码如下：

```
svg(filename=paste(root,"words.svg",sep=""),width = 40, height = 40)
plot(g.dir,layout=layout.fruchterman.reingold,
    vertex.label=label,vertex.label.cex=V(g.dir)$size/3,vertex.color=V(g.
dir)$color,vertex.label.family="GB1")
    dev.off()
```

重点词条的网络图如图 10-21 所示。

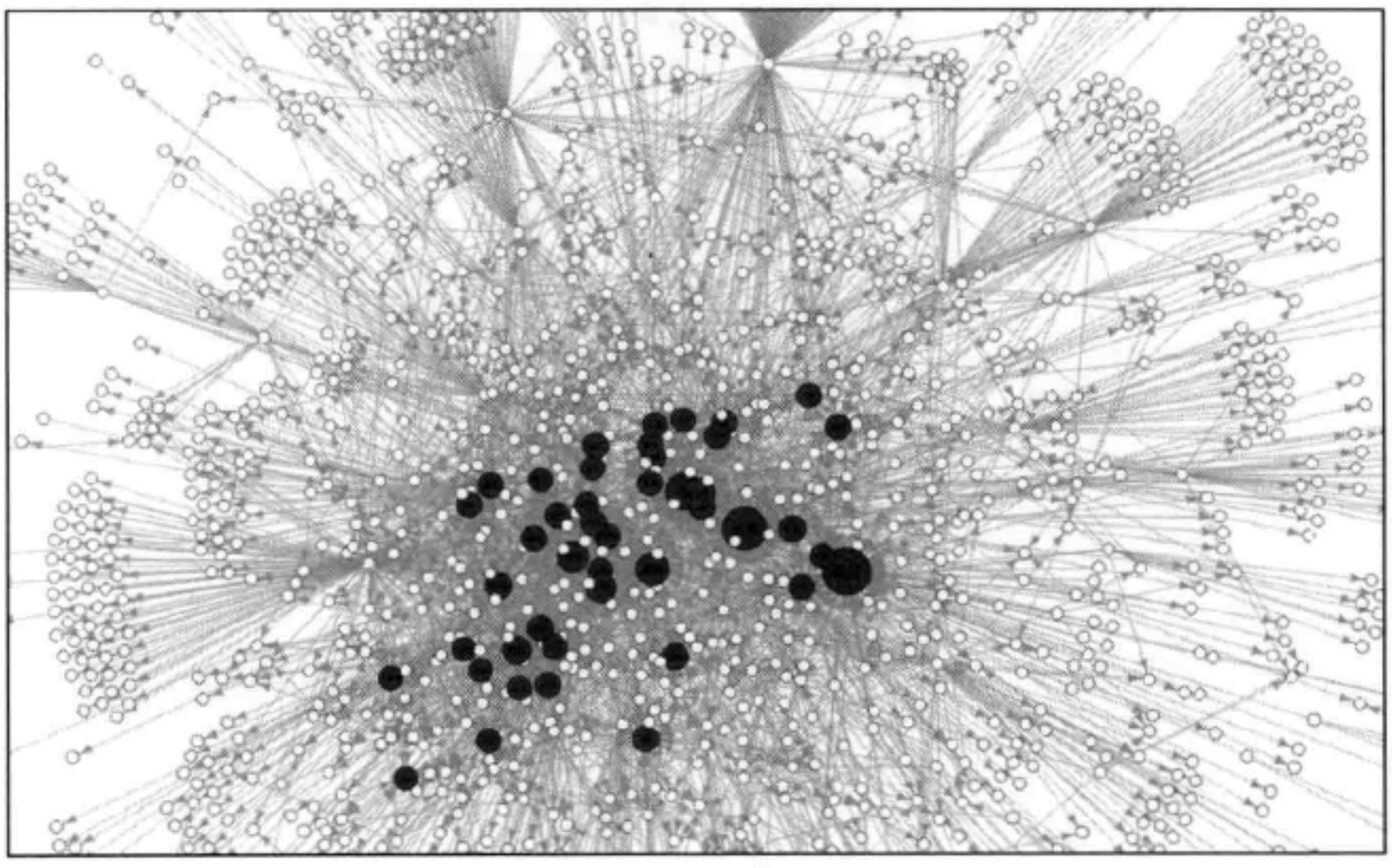


图 10-21 重点词条网络图

该图过大，读者可以下载数据集来显示上述代码的结果。

2. 绘制重点用户网络图

绘制步骤和上面的类似，只是这里使用点的出度来衡量重点用户，出度越大，表示该用户涉及的话题词条越多。这里使用“出度大于 20”作为筛选标准，并把重点用户的颜色置为绿色。代码如下：

```
std.degree.user=20
```

```
user.index<-(degree(g.dir,mode="out")>=std.degree.user)
user<-degree(g.dir,mode="out")[user.index]
names(user)<-V(g.dir)[user.index]$label
label=NA
label[user.index]<-names(user)
V(g.dir)$size=1# 一般词汇和用户的大小为 1
max.d<-max(user)
min.d<-min(user)
V(g.dir)[user.index]$size=2*(user-min.d)/(max.d-min.d)+2# 关键点词汇的大小为 2-4
V(g.dir)$color="white"
V(g.dir)[user.index]$color="green"
svg(filename=paste(root,"user.svg",sep=""),width = 40, height = 40)
plot(g.dir,layout=layout.fruchterman.reingold, vertex.label=label,vertex.label.
cex=V(g.dir)$size/3,vertex.color=V(g.dir)$color,vertex.label.family="GB1")
    dev.off()
```

绘制结果如图 10-22 所示。

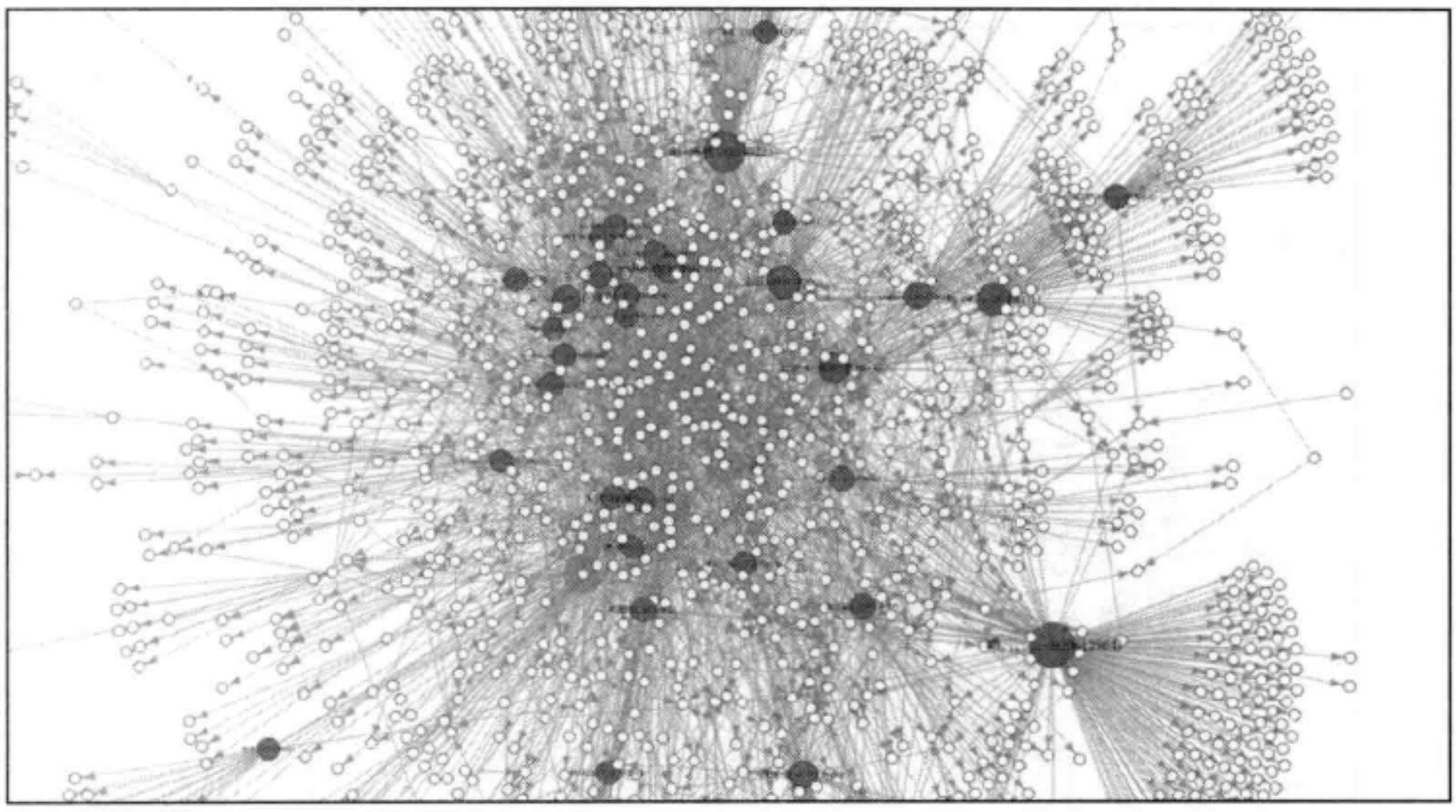


图 10-22 重点用户网络图

关于文本分析还有其他扩展知识，如 R 语言的网页爬取工具 XML 包可以自动爬取网页并分析出不同 div 等标签内的内容等。详见笔者的博客 http://www.bassary.com/?p=1107，这里就不一一讲解了。

10.3 XML 包爬取网页数据

R 语言的 XML 包可以方便地爬取网页数据，并使用 xpath 智能地读取某段 HTML 信息。

10.3.1 htmlParse 函数

htmlParse 函数是 XML 包加装 HTML 静态页面的函数。例如，要读取某电商网站的商品单页。代码如下：

```
url<-"http://product.dangdang.com/1459049401.html"
url.html<-htmlParse(url,encoding="UTF-8")
```

其中，url 是页面的 URL 地址。由于很多国内网站使用 gds 编码网页，所以这里要转化为 utf-8 编码（否则中文会有乱码情况出现），可通过把 htmlParse 函数的 encoding 参数设置为 UTF-8 来实现。获取的 url.html 结果代码如下：

```
    url.html
    <!DOCTYPE html PUBLIC" -//W3C//DTD XHTML 1.0 Transitional//EN"
" http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
    <meta charset="gbk">
    ...
            </style>
    </body>
    <!-- 249.107 -->
    </html>
```

10.3.2 getNodeSet 函数

上面已经获取了 HTML 信息，但是分析师一般只想要其中的部分信息，这时可以使用 getNodeSet 函数结合 xpath 来读取适当信息的节点 node。例如，由上面的 url.html 信息可知，商品价格的 HTML 代码如图 10-23 所示（使用 360 浏览器的开发人员工具展示的结果截图）。

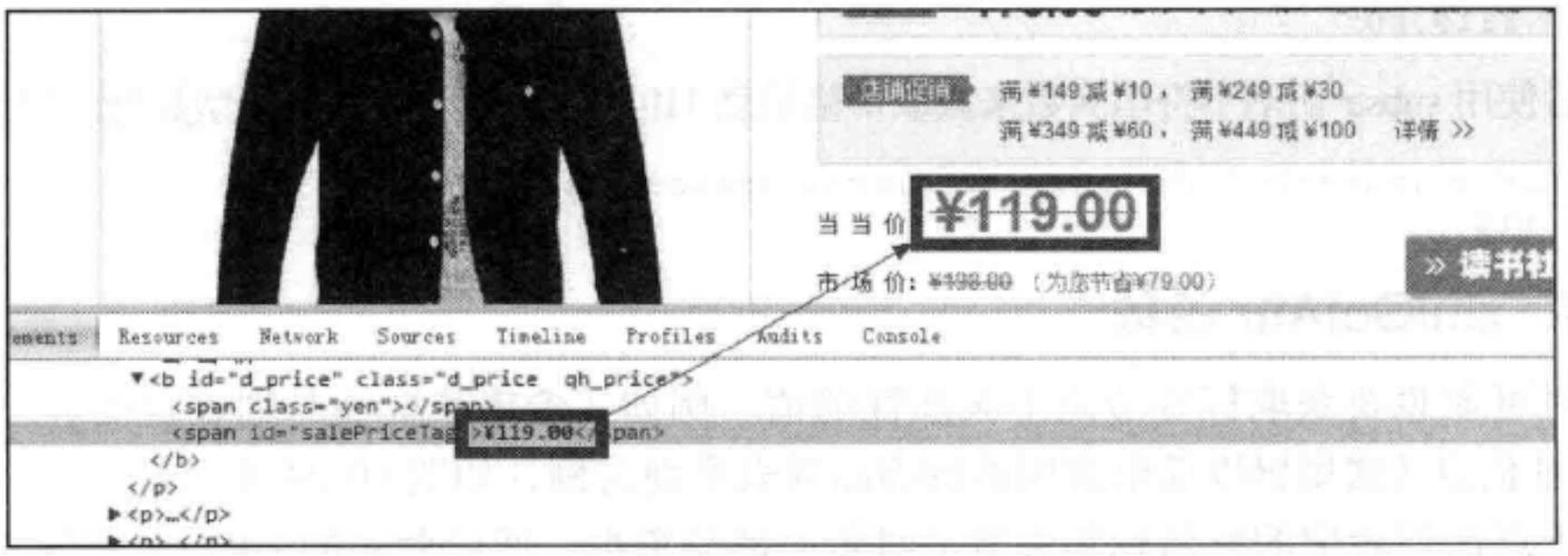


图 10-23 商品价格的 HTML 代码

可以看到，该商品的价格是 119 元（暂不爬取促销价格），代码展示￥119.00 是 标签内的内容，其 id 属性为“ salePriceTag”。在上层还有一个 标签，并含有 id 属性，为“d_price”，因此，可以使用如下代码读取价格信息的节点。

```
xpath<-"//*[@id='d_price']/span[@id='salePriceTag']"
price.node<-getNodeSet(url.html,xpath)
```

xpath 内的各个特殊字符如表 10-4 所示。

表 10-4 xpath 的特殊字符

字符	说明
//	任意个 HTML 嵌套标签
*	任意的 HTML 标签，实质这里是标签 ，即也可以使用 "//b[@id='d_price']/span[@id='salePriceTag']"
/	标识下一级标签，这里也可以使用 // 来表示下几级的标签。注意在 / 或 // 后必须添加一个标签或使用 * 代替任意标签
span[@id='salePriceTag']	细分标签，这里的 span[@id='salePriceTag'] 是指包含 id 属性为 salePriceTag 的 span 标签

price.node 最后返回一个 list 对象：

```
xpath
[[1]]
<span id="salePriceTag">¥119.00</span>
attr(,"class")
[1]" XMLNodeSet"
```

由于符合 "//*[@id='d_price']/span[@id='salePriceTag']" 条件的只有一处，所以 price.node 列表内只有一个元素。如果结果是多个，则可以增加 xpath 的搜索条件（设置增加更多级的嵌套标签或增加属性选择），直到 price.node 返回唯一的准确数值为止。

10.3.3 xmlValue 函数

通过 getNodeSet 函数得到价格信息节点 price.node 后，可以使用 xmlValue 函数读取节点下标签内的内容。代码如下：

```
price<- xmlValue(price.node [[1]]);price
[1]" ¥119.00"
```

然后使用 substr 切割字符串函数来截取价格信息 119.00，并转化为数值类型数据。代码如下：

```
price<-as.numeric(substr(price,2,nchar(price)));price
[1] 119
```

10.3.4 xmlGetAttr 函数

有时可能想要获取标签节点下某属性的值。例如，希望读取商品列表页面各个商品的 URL 地址信息（这里仍以某电商网站的商品列表页面为例，如图 10-24 所示）。

由于点击列表中的商品标题会自动跳转至商品单页，所以商品的 URL 地址信息实质是该标签 <a> 的 href 属性值，而标签 <a> 处于以下代码中：

```
<div class="shoplist">
     <ul class>
        <li name="lb" class="hover">
            <div class="inner">
                <a href=" 我们希望读取的 url 信息 ">
```

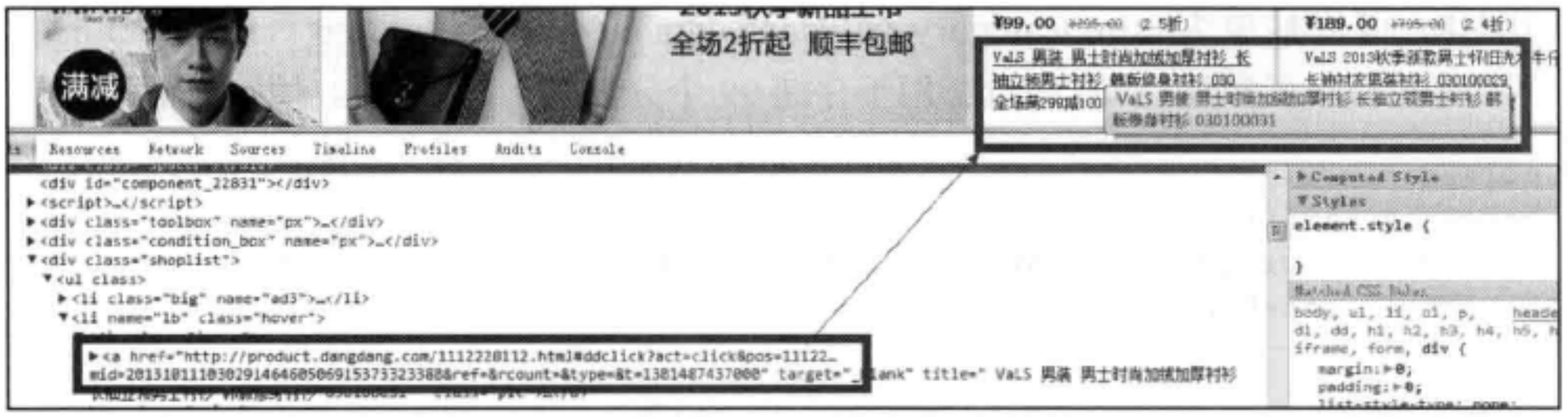


图 10-24 示例说明

因此，可以设置 xpath 为 //*[@class='shoplist']/ul/li[@name='lb']/div/a，最后的代码为：

```
url<-" http://category.dangdang.com/cid4008120-pg1.html"
url.html<-htmlParse(url,encoding="UTF-8")
xpath<-"//*[@class=' shoplist' ]/ul/li[@name=' lb' ]/div/a"
url.node<-getNodeSet(url.html,xpath)
```

返回的列表 price.node 包含 58 个元素（length(price.node) 等于 58）。其中第一个商品的 URL 节点信息为 url.node[[1]]。

通过 xmlGetAttr(<node 节点 >,< 属性名 >) 代码可以读取 node 节点内的某个属性值。例如，读取列表中第一个商品的 URL 节点中的 href 属性值。代码如下：

```
url<-xmlGetAttr(url.node[[1]],'href')
```

10.3.5 实例：爬取某电商网站衬衫类目商品的部分信息

本例中要爬取网站中衬衫类商品的商品 id、商品价格等信息。

1. 通过列表页读取所有商品的 URL 地址

图 10-25 是网站衬衫类目的列表页展示截图。



图 10-25 列表页展示截图

其URL可以抽象为http://category.dangdang.com/cid4008120-pg<分页序号>.html"。因此，可以使用for函数建立url向量，用于存储99个分页的URL地址。代码如下：

```
url<-c()
for(i in 1:99){
    url<-c(url,paste("http://category.dangdang.com/cid4008120-pg",i,".
    html",sep=""))
}
```

然后自定义一个read_url函数实现以下功能：读取99个列表页面内每个商品的URL地址。其核心代码可以参考10.3.4节中的内容，这里不再详细介绍。代码如下：

```
library(XML);
read_url<-function(url){
    url_vector<-c()
    i<-1
    for(i_url in url){
        i_url.html<-htmlParse(i_url,encoding="UTF-8")
        url.xpath<-getNodeSet(i_url.html,"//*[@class='shoplist']/ul/li[@
        name='lb']/div/a")
        url.i<-c()
        for(j in 1:length(url.xpath)){
            url.i<-c(url.i,xmlGetAttr(url.xpath[[j]],'href'))
        }
        url_vector<-c(url_vector,url.i)
        i<-i+1
    }
    url_vector
}
urls<-read_url(url)
```

2. 读取商品单页内部分信息

建立一个自定义函数read_xml，用于读取不同URL下的商品ID、商品价格等信息。读取代码如下：

```
read_xml<-function(url){
    id_vector<-c()# 商品id
    price_vector<-c()# 商品价格
    i<-1
    for(i_url in url){
        i_url.html<-htmlParse(i_url,encoding="UTF-8")
        #### 读取商品id####
        id.xpath<-getNodeSet(i_url.html,"//*[@id='prd_item_id']/i")
        id.i<-xmlValue(id.xpath[[1]])
        #### 读取商品价格 ####
        price.xpath<-getNodeSet(i_url.html,"//*[@id='salePriceTag']")
        price.i<-xmlValue(price.xpath[[1]])
        price.i<-substr(price.i,2,nchar(price.i))
        #### 存储向量 ####
```

```
        id_vector[i]<-id.i
        price_vector[i]<-as.numeric(price.i)
        i<-i+1
    }
    data.frame(id=id_vector,price=price_vector)
}
data<-read_xml(urls)
```

其核心代码可以参考前文，这里不细介绍，请读者自行研读。由于不同信息间读取的差异主要在 xpath 的设置上，所以读者可以扩展读取其他信息。这里的 urls 是通过 read_url 函数读取的 99 个列表页面内每个商品的 URL 地址。所以 data<-read_xml(urls) 可以输出由每个商品详情页内商品 ID 和价格组成的数据框。

第 11 章

网站指标监控系统的搭建

R 语言的 gWidgets 包提供了一个创建 GUI 的统一框架，这给利用 R 语言搭建网站数据指标的监控系统提供了基础。

11.1 gWidgets 包基础

11.1.1 环境搭建

使用 gWidgets 包建立 GUI 的基本原理如下（这里以 Windows 操作系统为例进行说明）：

因为要通过 gWidgetsRGtk2 包来解析 R 环境中用于建立 GUI 窗口的代码，并调用 Windows 系统下的 Gtk 插件并显示 GUI 界面。所以在开始编码前要保证已经安装了如下环境：

1）安装 2.9.2 版本的 R 语言环境。

2）安装 R 语言环境下的 gWidgetsRGtk2 扩展包，可以在 R 环境下输入代码：

```
install.packages("gWidgetsRGtk2", dep=TRUE)
```

3）安装 Windows 操作系统下 2.12.9 版本的 Gtk 插件，可以打开网址 http://sourceforge.net/projects/gladewin32/files/gtk%2B-win32-runtime /2.12.9/ ，并点击 gtk-2.12.9-win32-2.exe 下载，如图 11-1 所示。

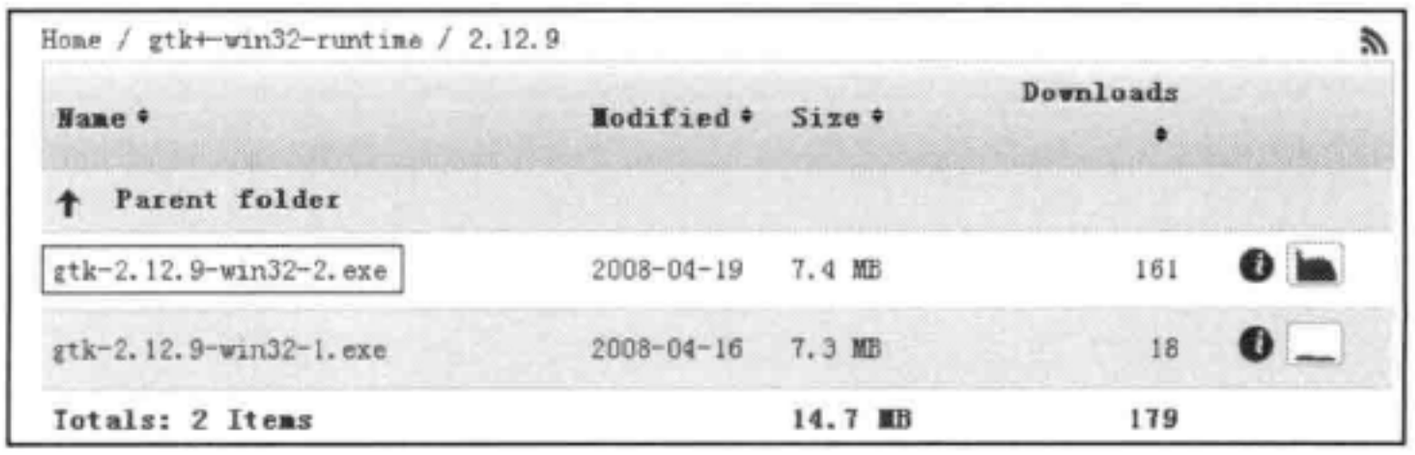


图 11-1　gtk 下载页面

下载后点击源文件进行安装。

上述各软件的版本如表 11-1 所示。

表 11-1 版本信息

软件和扩展包	版本号
R 环境	2.9.2
gWidgetsRGtk2 包	安装扩展包时自动匹配
Gtk 插件	2.12.9

对于其余的版本，笔者无法保证可以搭配成功。

11.1.2 引例

先给出一个案例，以便引出 GUI 编程中的主要任务和概念。该案例用于展示某交友网站用户活动的数量，可以查看以月和周为周期的数据，并按照比例和数据两种方式展示，最终的展示效果如图 11-2 所示。

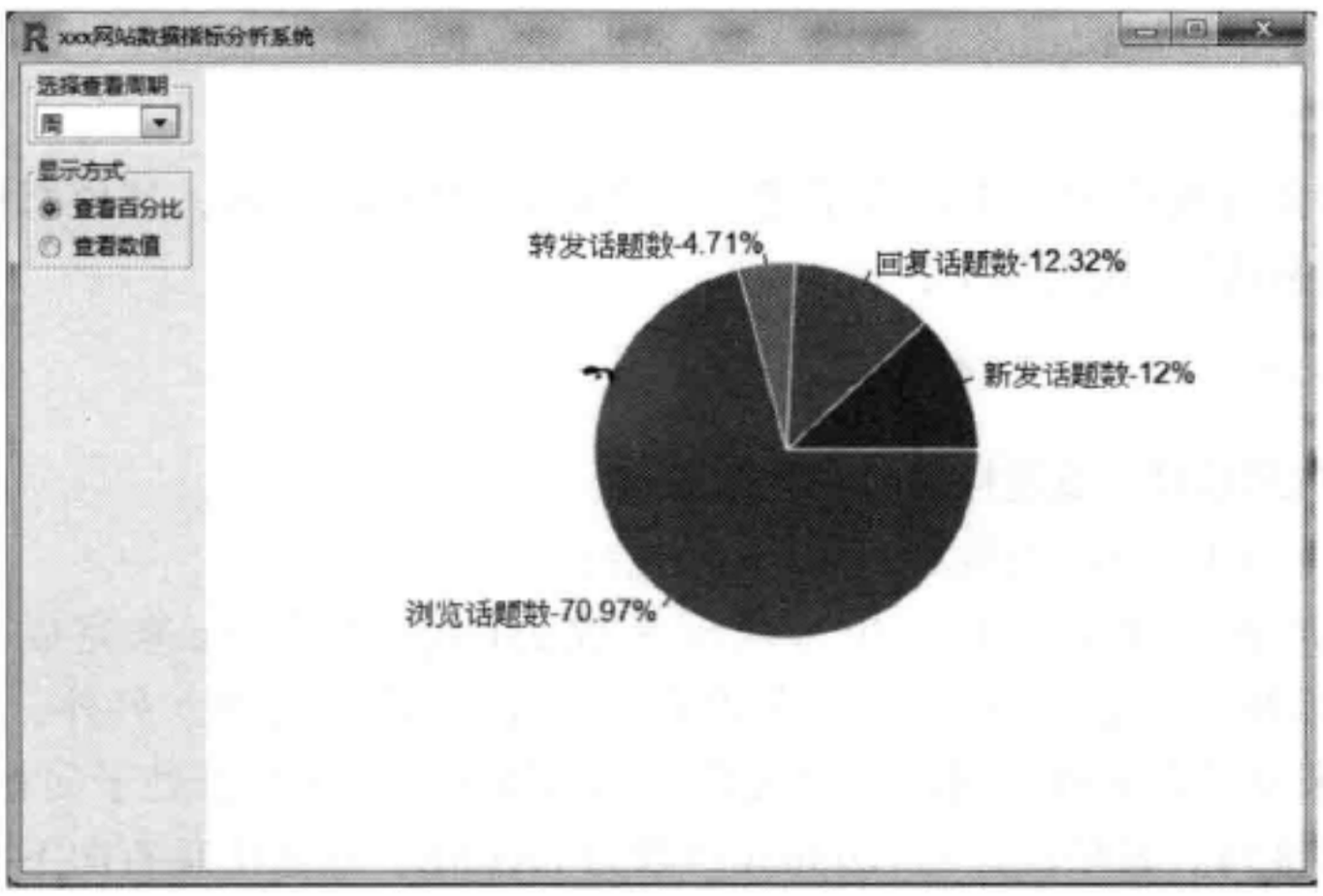


图 11-2 饼图

其中涉及的主要概念如表 11-2 所示。

表 11-2 涉及概念

概念	说明
窗口	用于承载 GUI 内各控件的最基本部件，使用 gwindow 函数建立
布局框（容器）	用于布局控件的基础部件，使用 ggroup 或 gframe 函数建立
控件	接收用户点击等操作的页面元素，包括的控件如下： 1）“选择查看周期”下拉菜单 2）“显示方式”单选按钮 3）用于展示饼状图的图形展示区域
触发函数	用户操控控件后自动触发的 R 语言函数，用于更新 GUI 界面内的数据展示，实现人机交互

实现上述界面的代码主要完成如下操作。

1. 加装相关扩展包

这里主要是加载 gWidgetsRGtk2 包，其代码如下：

```
options(guiToolkit = "RGtk2")
library(gWidgets)
```

如果有错误提示，则表明软件的版本可能有问题或环境没有搭建好。另外，在后续代码中需要使用到颜色包 RColorBrewer，其加载代码如下：

```
library(RColorBrewer)
```

2. 准备原始数据

需要查看的是月周期数据和周周期数据，这里准备的原始数据如下：

```
mouth<-c(36313,35500,16965,230116)
week<-c(8646,8875,3393,51137)
```

在实际操控中，可以使用 R 语言连接 mySQL 等数据库直接动态读取。

3. 建立窗口

所有控件均要存放于窗口中，这里建立一个窗口对象 window，其顶部标题为“ xxx 网站数据指标分析系统”，代码如下：

```
window = gwindow("xxx 网站数据指标分析系统 ")
```

4. 建立并布局控件，设置控件中的触发函数

此处主要控制 GUI 界面的展示，其任务包括：

1）设置布局框。这里主要使用布局框函数 ggroup 和 gframe 来完成，其中，gframe 可以设置顶部标题，而 ggroup 只是一个没有任何展示的布局框。另外，要通过 cont(或 container) 参数来设置布局框之间的嵌套关系。默认情况下，布局框处于父布局框的左上角，并依次水平向右排列。如果设置 horizontal 参数为 FALSE，则依次垂直向下排列。布局框的宽度依据内部控件大小自适应，如果设置 expand 参数为 T，则布局框扩充全部，使用 size 函数也可以直接设置长和宽。实例中的实施代码如下：

```
BigGroup = ggroup(cont = window)
    group=ggroup(horizontal=FALSE,container=BigGroup)# 建立左边的设置区域
        period.gf=gframe(" 选择查看周期 ",container=group)
        view.gf=gframe(" 显示方式 ",container=group)
    group.plot=ggroup(container=BigGroup,expand=T)# 建立左边的设置区域
```

结果如图 11-3 所示。

其中，BigGroup 是最底层的布局框，它与窗口一样大。还设置了左侧和右侧两个虚拟的布局框 group 和 group.plot。由于左侧的布局框 group 还要垂直地放置“查看周期”和“显示方式”两个布局框，所以它的 horizontal 参数被设置为 FALSE。右侧只用于展示饼状图，

其宽度设置为自动填充全部（设置 expand 参数为 T）。因为“查看周期”和“显示方式”两个布局框的顶部要展示标题，所以使用 gframe 函数来建立。并设置 container 参数为 group，使其作为 group 布局框的子布局框。

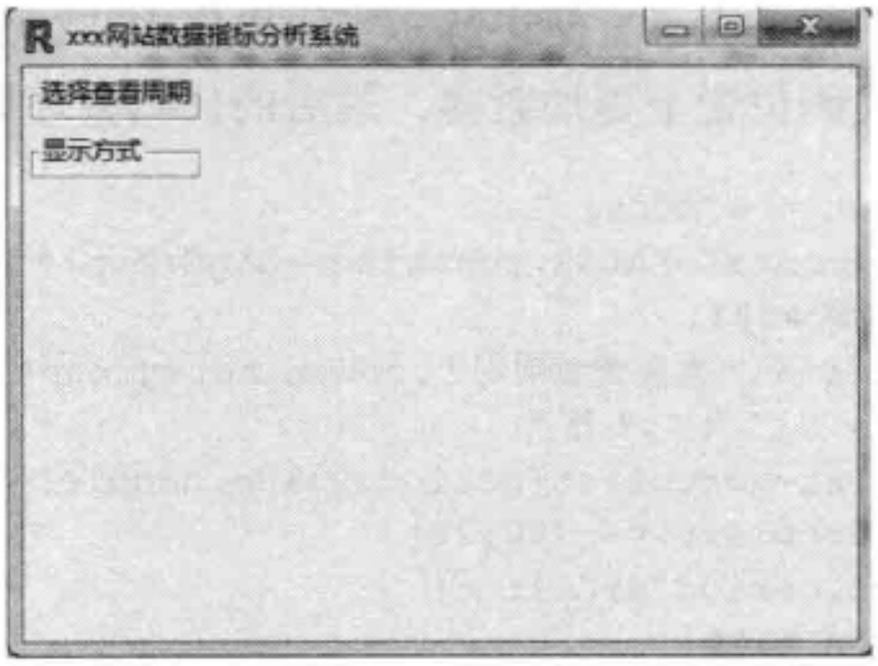


图 11-3 结果展示图 1

2）建立控件，设置控件的输入值，并设置触发函数。在这个步骤中，先建立数据查看周期的下拉列表框对象 period.droplist，并设置可选值为“周”和“月”。代码如下：

```
period.value= c("周","月")
period.droplist=gdroplist(period.value,handler=updatePlot)
size(period.droplist)<-c(80,20)
```

其中，handler 参数用于设置该布局的触发函数，可设置为自定义函数 updatePlot。然后设置控件的宽为 80，高为 20。代码如下：

```
size(period.droplist)<-c(80,20)
```

显示方式的单选按钮控件的创建方式与之类似，代码如下：

```
view.value= c("查看百分比","查看数值")
view.radio=gradio(view.value,horizontal=FALSE,handler=updatePlot)
```

由于“查看百分比”和“查看数值”的两个单选按钮是垂直排列的，所以 gradio 控件函数的 horizontal 参数被设置为 FALSE。

3）存放控件于相应布局框中。以上只是完成了控件的建立和触发函数，接下来要使用 add(<布局框>,<控件>) 把控件放于相关布局框的位置上。上面建立的 period.gf、view.gf 以及 group.plot 布局框分别用于放置下拉列表框控件 period.droplist、单选按钮控件 view.radio 以及图形展示区，相关代码如下：

```
add(period.gf,period.droplist)
add(view.gf,view.radio)
add(group.plot,ggraphics())# 建立右边的显示区域
```

注意，由于图形展示区无特殊参数，因此直接使用函数 ggraphics() 建立并放于 group.

plot 中。或者在建立控件函数中直接设置 cont= 相关布局框。代码如下：

```
period.droplist=gdroplist(period.value,handler=updatePlot,cont=period.gf)
view.radio=gradio(view.value,horizontal=FALSE,handler=updatePlot,cont=view.gf)
```

上述代码的编写顺序主要是为了方便讲解，实际操作中往往使用不同的布局控件来分隔代码，使控件和布局框在代码位置上更加紧凑，最后的代码整理如下：

```
BigGroup = ggroup(cont = window)
    group=ggroup(horizontal=FALSE,container=BigGroup)# 建立左边的设置区域
        #### 添加查看周期 ####
        period.gf=gframe(" 选择查看周期 ",container=group)
        period.value= c(" 周 "," 月 ")
        period.droplist=gdroplist(period.value,handler=updatePlot)
        size(period.droplist)<-c(80,20)
        add(period.gf,period.droplist)
        #### 添加显示方式 ####
        view.gf=gframe(" 显示方式 ",container=group)
        view.value= c(" 查看百分比 "," 查看数值 ")
        view.radio=gradio(view.value,horizontal=FALSE,handler=updatePlot)
        add(view.gf,view.radio)
  group.plot=ggroup(container=BigGroup,expand=T)# 建立左边的设置区域
        add(group.plot,ggraphics())# 建立右边的显示区域
```

结果如图 11-4 所示。

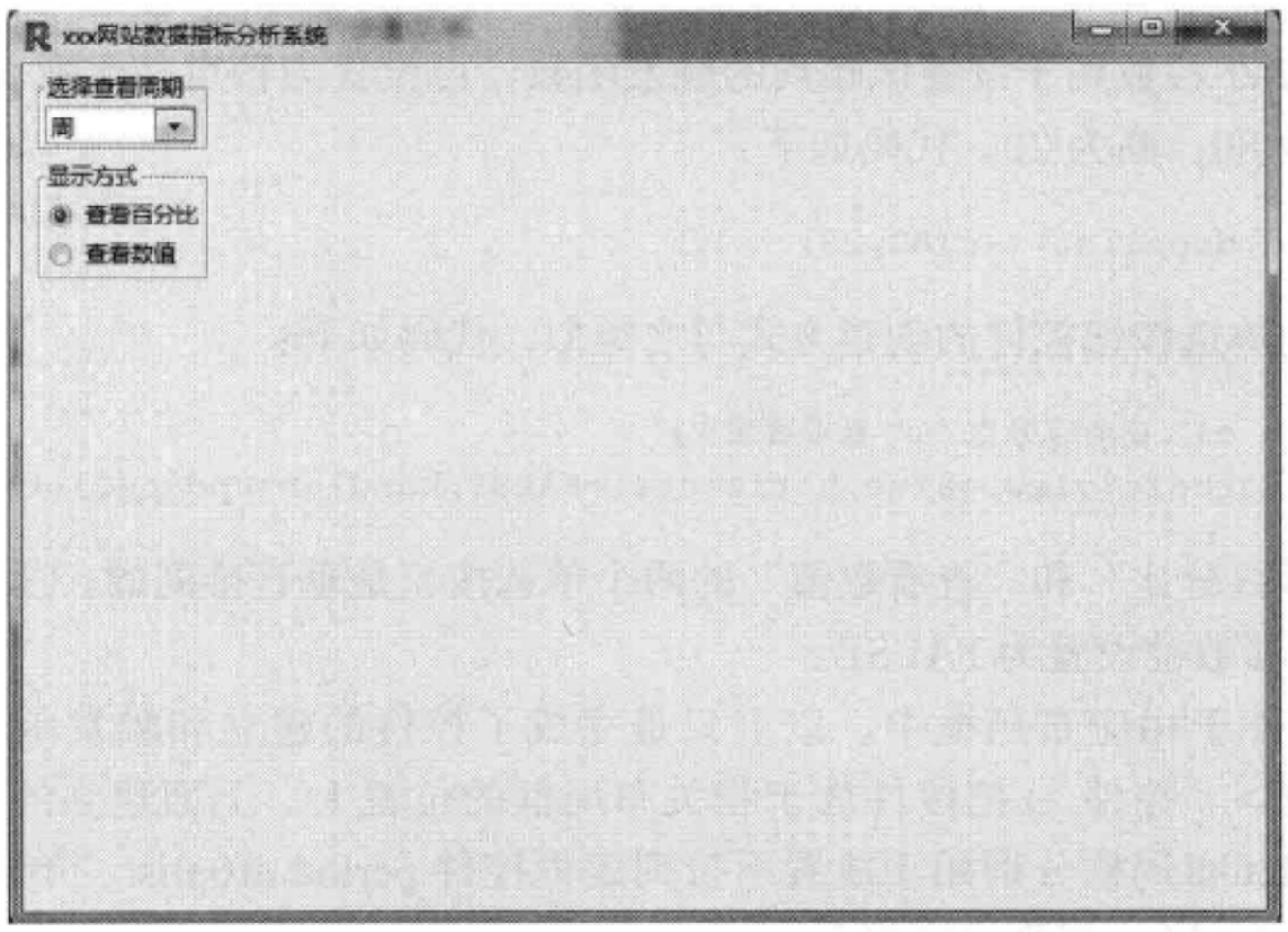


图 11-4 结果展示图 2

5. 定义控件的触发函数

在建立控件时要设置相关的触发函数，上文使用的是自定义函数 updatePlot，这里给出完整的代码：

```
updatePlot<-function(h,...){
    x.name<-c()
    period.num<-which(period.value==svalue(period.droplist))
    if(period.num==1){
        do.num<-week
    }else{
        do.num<-mouth
    }
    names(do.num)<-c("新发话题数","回复话题数","转发话题数","浏览话题数")
    view.num<-which(view.value==iconv(svalue(view.radio),"UTF-8","gbk"))
    if(view.num==1){
        for(i in 1:4){
            x.name[i]<-paste(names(do.num)[i],"-",
                             round(10000*do.num[i]/sum(do.num))/100,"%",sep="")
        }
    }else{
        for(i in 1:4){
            x.name[i]<-paste(names(do.num)[i],"-",do.num[i],sep="")
        }
    }
    pie(do.num,labels=x.name,col=brewer.pal(4,"Set1"),border="white",family="GB1")
}
```

触发函数的参数可以统一设置为 function(h,...)，其内部使用的数据主要有两个来源：

1）全局对象，如 week 和 mouth。

2）控件的当前值，如 svalue(period.droplist) 和 svalue(view.radio)。其中 svalue 是读取控件当前值的函数。

由于下拉列表框 period.droplist 的值为 period.value= c("周","月")，所以如果用户在下拉列表框中选择了“周”，则经 period.num<-which(period.value==svalue(period.droplist)) 代码处理后，period.num 返回 1，如果选择“月”，则返回 2。最后，通过一个选择语句 if-else 把 period.num 等于 1 时真正使用的内部数据 do.num 设置为 week（周数据），否则 do.num 会设置为 mouth（月数据）。

单选按钮 view.radio 的处理模式与下拉列表框 period.droplist 类似。只是 svalue 函数对读取单选按钮的中文会存在编码问题，默认返回 gbk 编码，因此需要使用 iconv 函数转化为 utf-8 编码：

```
view.num<-which(view.value==iconv(svalue(view.radio),"UTF-8","gbk"))
```

另外，如果用户选择“查看百分比”，则 view.num 等于 1，如果选择“查看数值”，则 view.num 等于 2。

最后，使用 pie 函数以饼状图形式展示 do.num 的数据或百分比。

注意

由于在建立控件时需要设置触发函数，所以在实际的代码中，创建触发函数的代码需要放于使用相关控件建立代码的前面。这里只是为了讲解方便才放在后边的。

6. 初始化

控件 period.droplist 和 view.radio 的默认值（打开窗口后）是其取值向量 period.value[1] 和 view.value[1]，即“周”和“查看百分比”。自动运行触发函数 updatePlot 展示图形区域的初始化内容，代码如下：

```
updatePlot()
```

最终结果如图 11-5 所示。

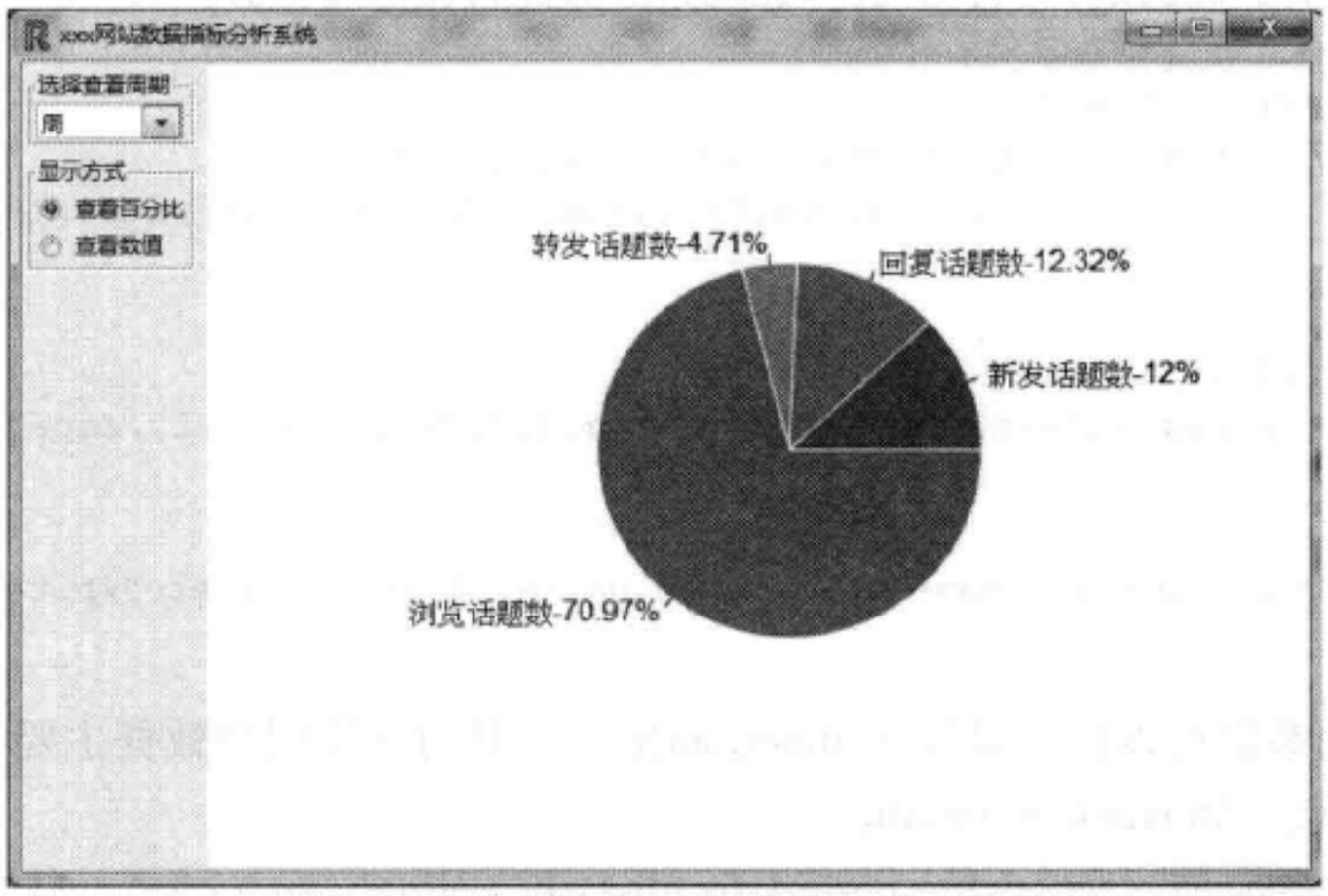


图 11-5 结果展示图 3

11.1.3 常用控件

上面大致讲解了下拉列表框和单选按钮的使用方法，下面详细讲解常用控件的相关内容。

1. 信息提示框 gmessage

建立一个错误提示框，如果用户不点击确定按钮则无法进行下一步操作。代码如下：

```
gmessage(" 输入错误，请修改信息 ", title="gmessage",icon="error")
```

结果如图 11-6 所示。

这里的 icon 用于设置左上角的图标，可以不设置而使用默认图标。



图 11-6 信息提示框示例

2. 确认框 gconfirm

创建一个确认框，如果点击“确定”按钮，则在终端显示“ok”，如果点击“取消”按钮，则不做任何操作。代码如下：

```
gconfirm(" 是否在终端输出 ok？ ", title="gconfirm",handler= function(h,...) print("ok"))
```

结果如图 11-7 所示。

注意，这里的触发函数比较简单，可以直接写于 handler 中：

```
function(h,...) print("ok")
```

3. 输入框 ginput

创建一个输入框，提示语为“请输入置信度”，默认输入值为 0.95。代码如下：

```
g.config <-ginput(" 请输入置信度 ",text="0.95",title=" 注意 ",icon="info")
```

结果如图 11-8 所示。



图 11-7　确认框示例

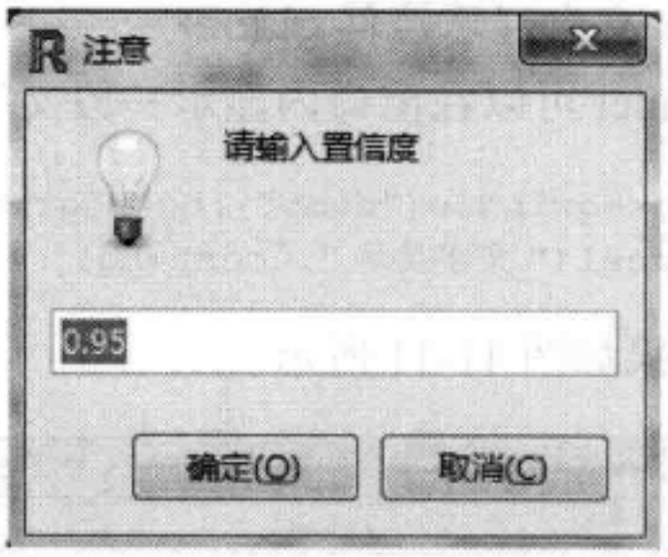


图 11-8　输入框示例

4. 文件浏览框

使用代码 source(gfile()) 可以弹出文件浏览框，默认浏览路径为此时的工作路径，如图 11-9 所示。

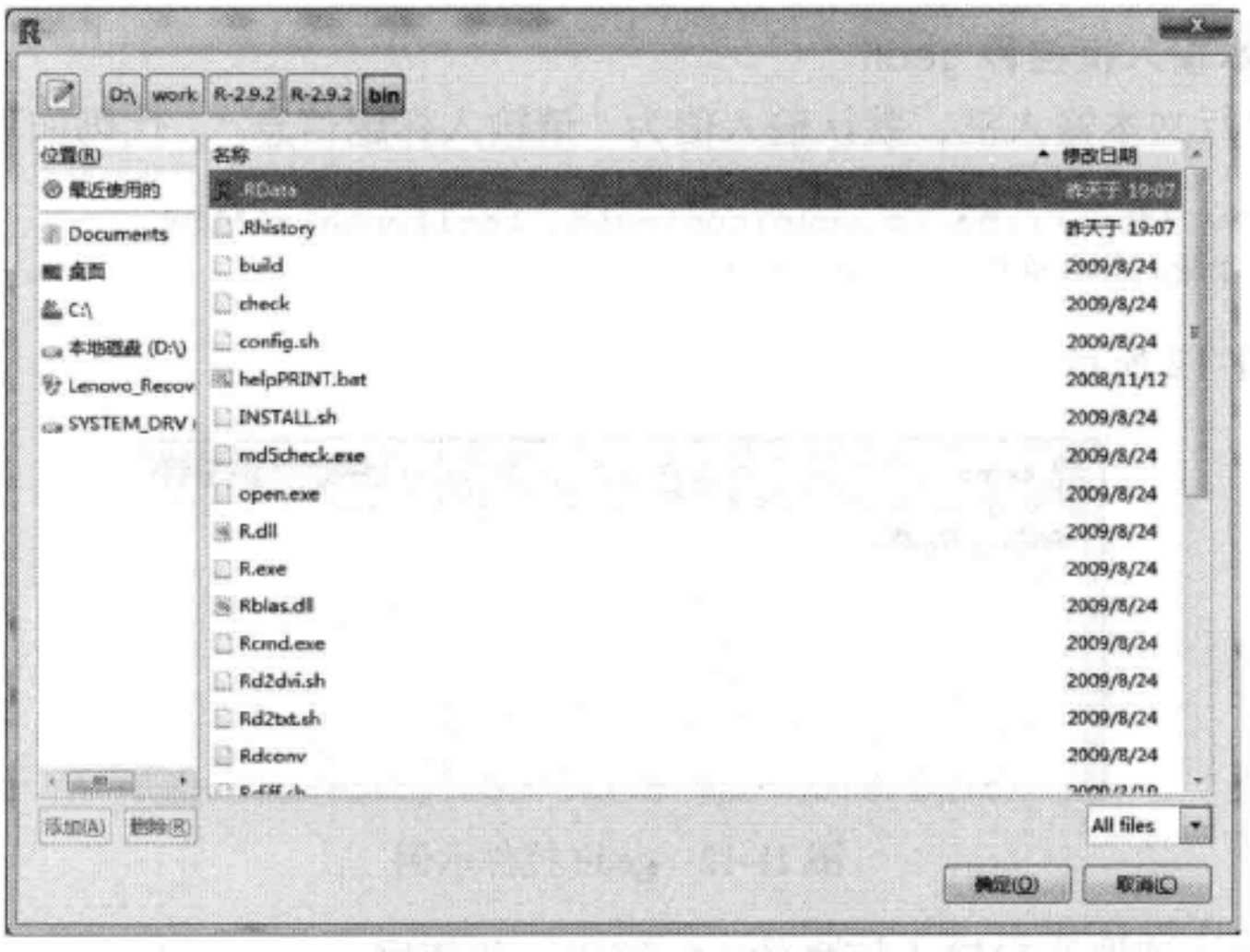


图 11-9　文件浏览框示例

选择其中的某个 R 文件后，系统自动运行命令 source< 文件 >，注意尽量不要在路径中出现中文。

以上讲解了独立于窗口的几种弹出框，下面讲解只能存在于窗口（或布局框）内的控件。

5. 按钮控件 gbutton

在窗口 win 中建立一个按钮，按钮标题是“确定”。代码如下：

```
win<-gwindow("demo");bg<-ggroup(cont=win, horizontal=FALSE)
gbutton(" 确定 ", cont=bg)
```

结果如图 11-10 所示。

6. 文字展示控件 glabel

glabel 可以在窗口内显示一段文字，代码如下：

```
win<-gwindow("demo");bg<-ggroup(cont=win, horizontal=FALSE)
glabel(" 文字展示 ", cont=bg)
```

结果如图 11-11 所示。

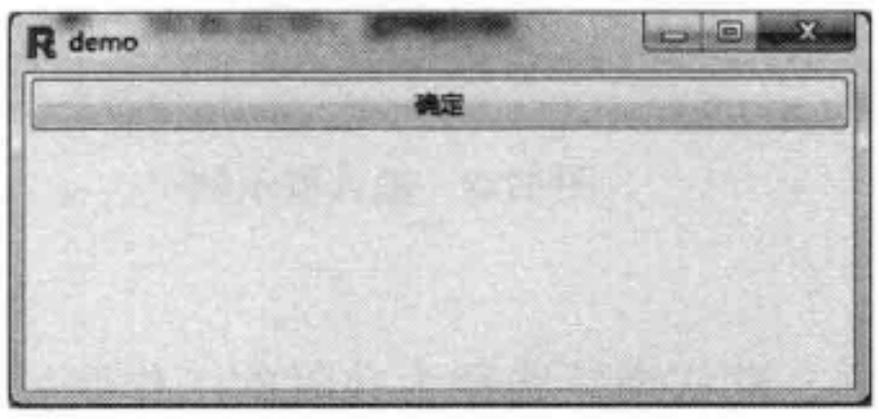


图 11-10 gbutton 控件示例



图 11-11 glabel 控件示例

7. 单行文本输入框控件 gedit

建立一个单行文本输入框，默认输入值为 " 请输入标题信息 "。代码如下：

```
win<-gwindow("demo");bg<-ggroup(cont=win, horizontal=FALSE)
g<-gedit(" 请输入标题信息 ", cont=bg)
```

结果如图 11-12 所示。

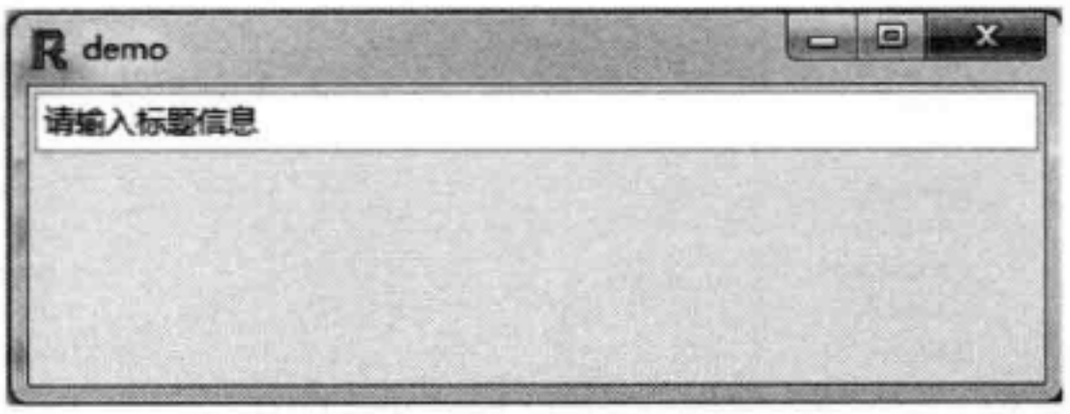


图 11-12 gedit 控件示例

使用 svalue(g) 读取当前输入信息的 gds 编码，并使用 iconv(svalue(g),"UTF-8","gbk") 将

其转化为 utf-8 编码。代码如下：

```
iconv(svalue(g),"UTF-8","gbk")
[1] "请输入标题信息"
```

8. 多行文本输入框控件 gtext

建立一个多行文本输入框，默认输入 3 行文字。代码如下：

```
win<-gwindow("demo");bg<-ggroup(cont=win, horizontal=FALSE)
g<-gtext("请输入标题信息 \n 第二行 \n 第三行", cont=bg)
```

结果如图 11-13 所示。

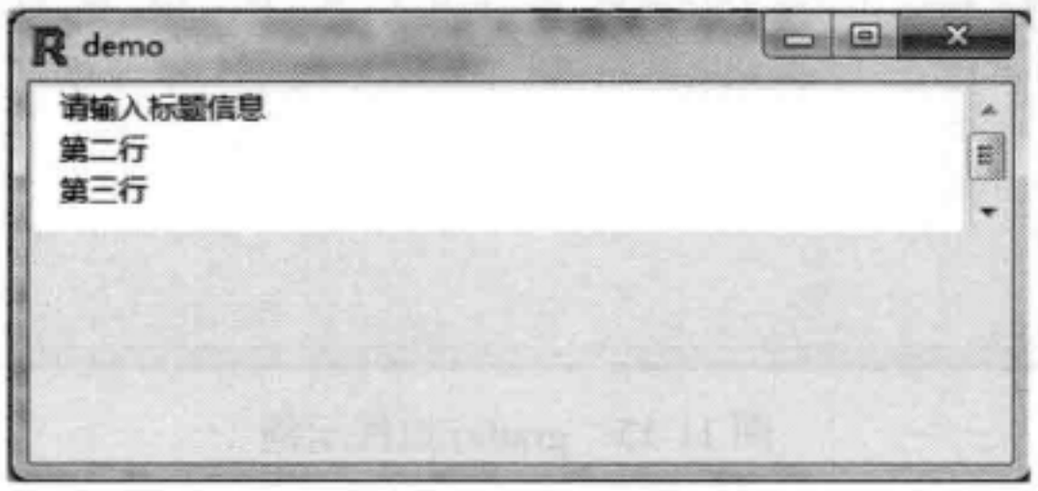


图 11-13　gtext 控件示例

与 gedit 控件一样，可使用如下代码来读取输入文字：

```
iconv(svalue(g),"UTF-8","gbk")
[1] "请输入标题信息 \n 第二行 \n 第三行 \n"
```

9. 复选框控件 gcheckbox

建立一个复选框，标题为“是否使用?”。代码如下：

```
win<-gwindow("demo");bg<-ggroup(cont=win, horizontal=FALSE)
g<-gcheckbox("是否使用？ ", cont=bg)
```

结果如图 11-14 所示。

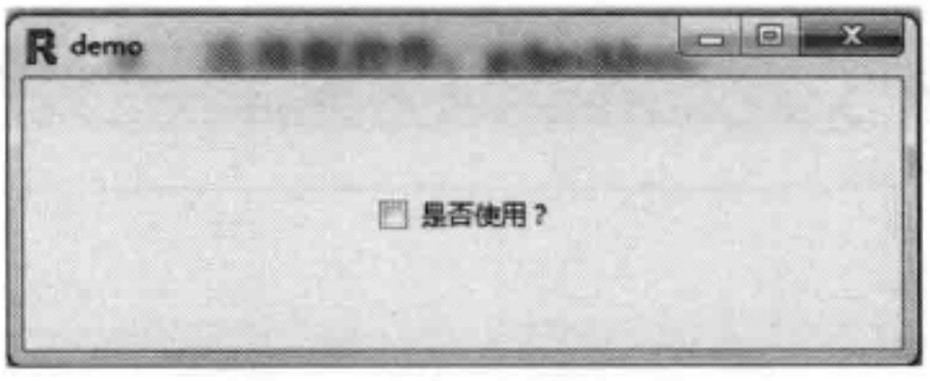


图 11-14　gcheckbox 控件示例

默认情况下是未选中，通过设置 checked 参数为 TRUE 可以确定为默认选中。

使用 svalue(g) 返回控件是否被选中，如果未选中，则返回 FALSE，否则返回 TRUE。代码如下：

```
svalue(g)
[1] FALSE
```

10. 单选按钮组控件 gradio

建立一个单选按钮组控件，其中的单选项依次为“好”“中”“差”，默认选中“好”。

```
win<-gwindow("demo");bg<-ggroup(cont=win, horizontal=FALSE)
g.value= c("好","中","差")
g=gradio(g.value,horizontal=FALSE,cont=bg)
```

结果如图 11-15 所示。

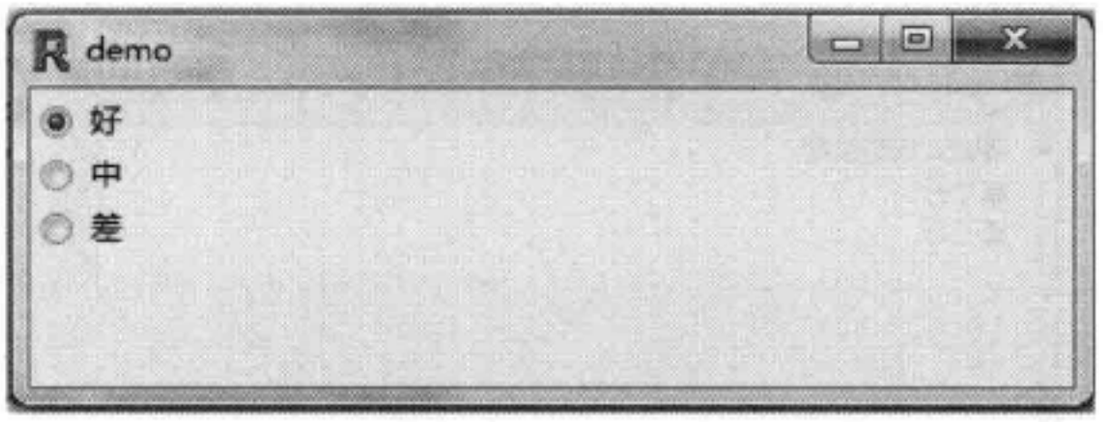


图 11-15　gradio 控件示例

如果想把 3 个选项水平排列，则设置 horizontal 为 TRUE（默认取值）。

与 gedit 控件一样，可使用如下代码来读取现在选择的文字。

```
iconv(svalue(g),"UTF-8","gbk")
 [1] "好"
```

11. 下拉列表框控件 gdroplist

下拉列表框和单选按钮只是展示形式上有些不同，其建立方式都一样的，代码如下：

```
win<-gwindow("demo");bg<-ggroup(cont=win, horizontal=FALSE)
g.value= c("好","中","差")
g=gdroplist(g.value,horizontal=FALSE,cont=bg)
```

结果如图 11-16 所示。

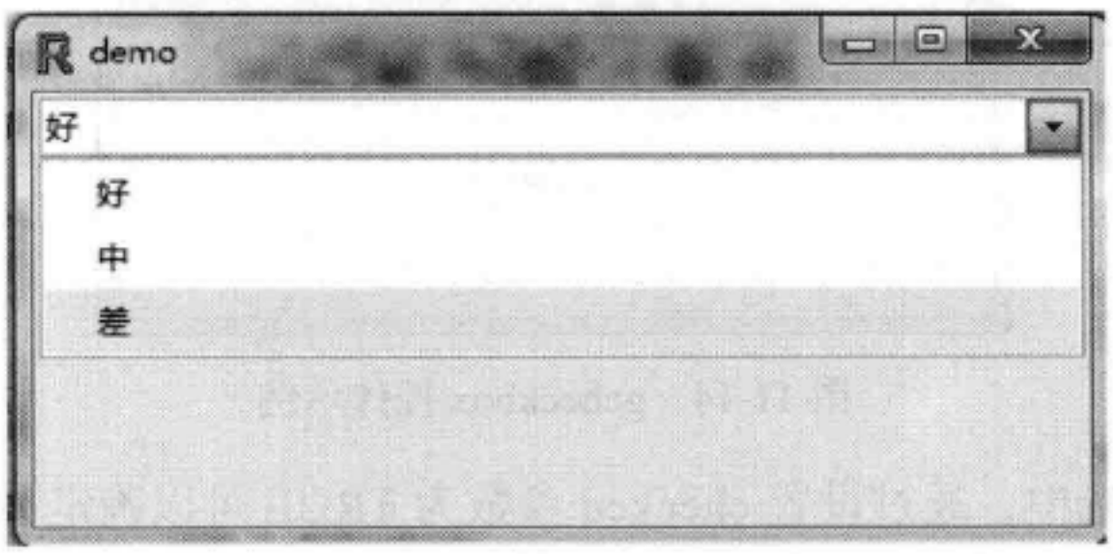


图 11-16　gdroplist 控件示例

使用 svalue(g) 可以直接返回 utf-8 编码的选择文字，不必再使用 iconv 函数转换编码，代码如下：

```
svalue(g)
[1]"好"
```

12. 复选框组控件 gcheckboxgroup

复选框组的建立与单选按钮组类似，可使用 gcheckboxgroup 实现，代码如下：

```
win<-gwindow("demo");bg<-ggroup(cont=win, horizontal=FALSE)
g.value= c("好","中","差")
g=gcheckboxgroup (g.value,horizontal=FALSE,cont=bg)
```

结果如图 11-17 所示。

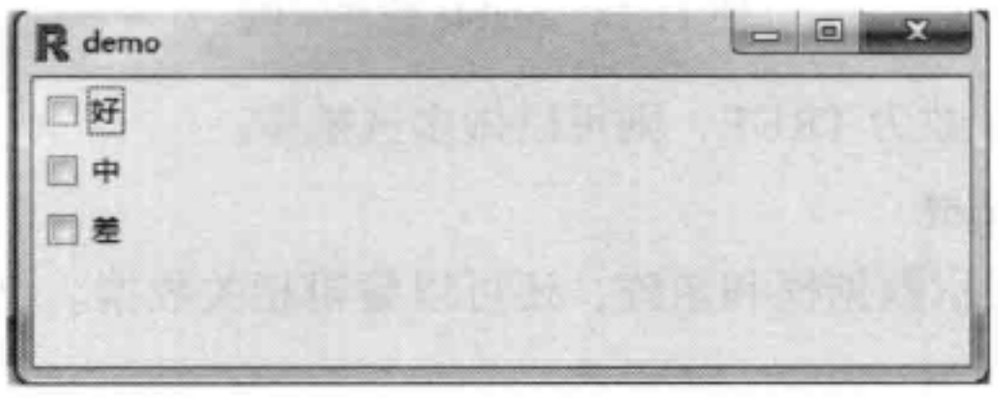


图 11-17　gcheckboxgroup 控件示例

使用 svalue(g) 返回选择的数据项，如果未选择，则返回 character(0)，否则返回向量。代码如下：

```
svalue(g)
[1] "好" "中"
```

13. 数据表格控件 gtable

gtable 用于以表格形式展示数据框或矩阵对象，并可以选择其中的数据。例如，数据框 data 的内容如下：

```
data
         date     pv      v     uv     ip
1  2013.01.14 532251 389721 223065 194284
2  2013.01.15 574540 398998 216166 188711
                <中间省略>
13 2013.01.26 677159 454841 280163 249604
14 2013.01.27 635077 436527 269671 239499
```

可使用如下代码展示 data 中的内容：

```
win<-gwindow("demo");bg<-ggroup(cont=win, horizontal=FALSE)
g=gtable(data,cont=bg)
size(g)<-c(600,400)
```

结果如图 11-18 所示。

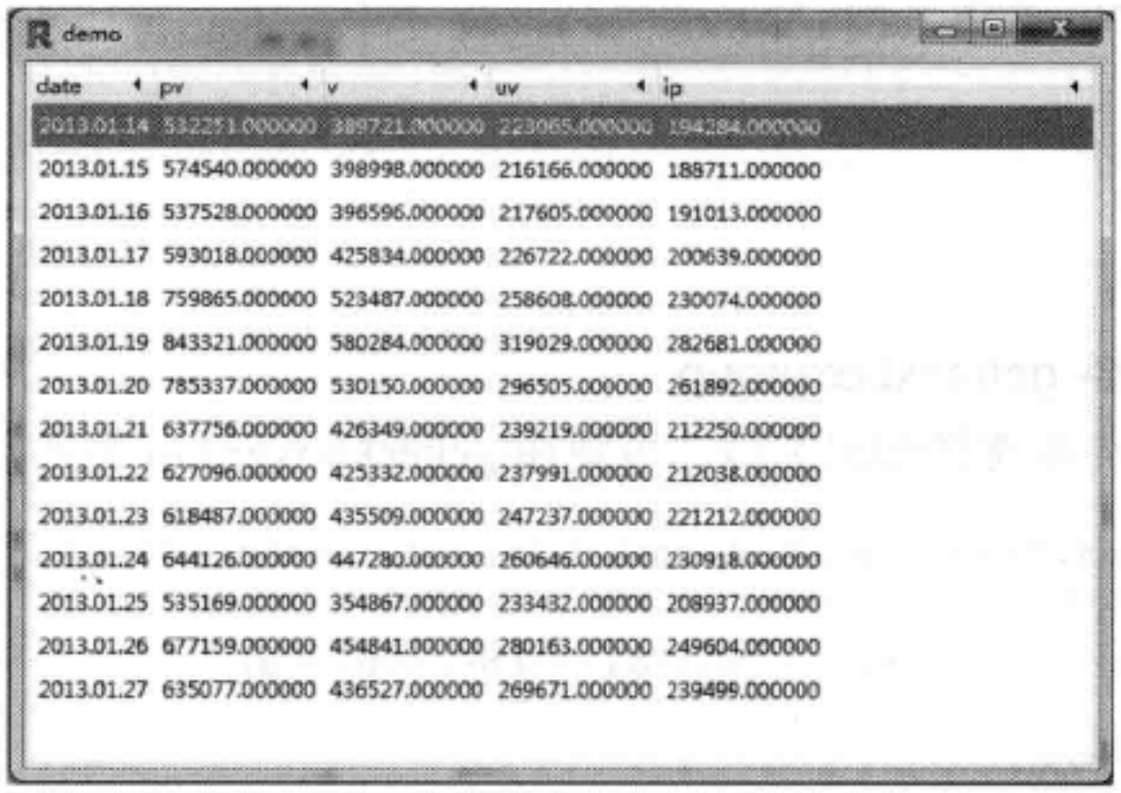


图 11-18　gtable 控件示例

如果设置 multiple 参数为 TRUE，则可以做多选操作。

14. 数据表格控件 gdf

gdf 控件不仅可以展示数据框和矩阵，还可以编辑相关数据。仍使用上面实例中的数据框 data 为例。代码如下：

```
win<-gwindow("demo");bg<-ggroup(cont=win, horizontal=FALSE)
g=gdf(data,cont=bg)
size(g)<-c(600,400)
```

结果如图 11-19 所示。

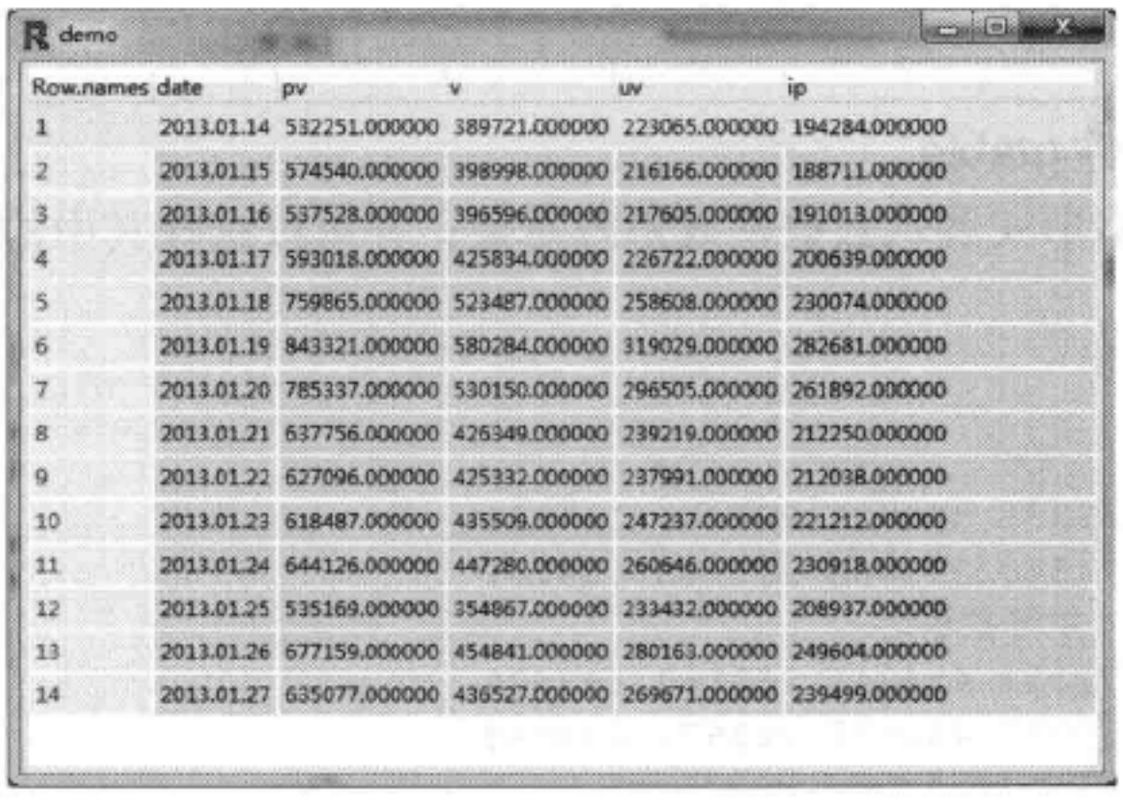


图 11-19　gdf 控件示例

15. 连续数据的滑动条控件 gslider

使用如下代码可以设置 0 ～ 100 的连续数据。

```
win<-gwindow("demo");bg<-ggroup(cont=win, horizontal=FALSE)
```

```
g<-gslider(from=0, to = 100, by = 1,cont=bg)
```

结果如图 11-20 所示。

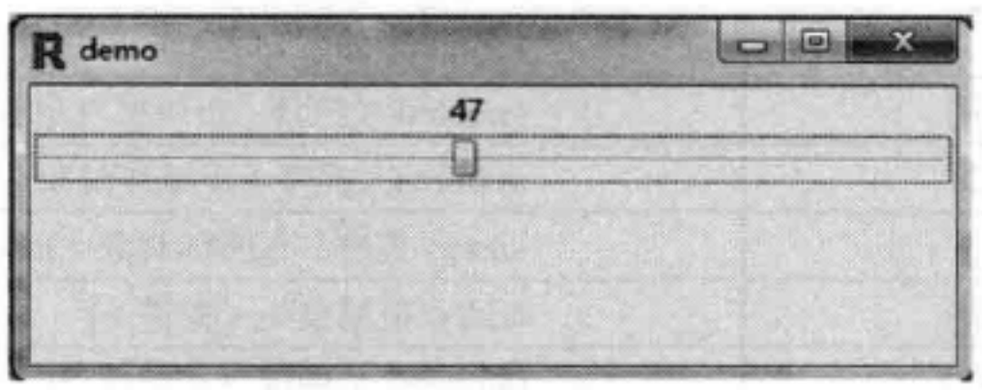


图 11-20 gslider 控件示例

这里显示的 47 是用户调整滑动条后的数值，可以使用 svalue(g) 直接读取。

16. 连续数据的输入控件 gspinbutton

使用 gspinbutton 控件可以直接设置数值，并且可使用上下按钮自动增减间距值。代码如下：

```
win<-gwindow("demo");bg<-ggroup(cont=win, horizontal=FALSE)
g<-gspinbutton(from=0, to = 1, by = 0.1,cont=bg)
```

结果如图 11-21 所示。

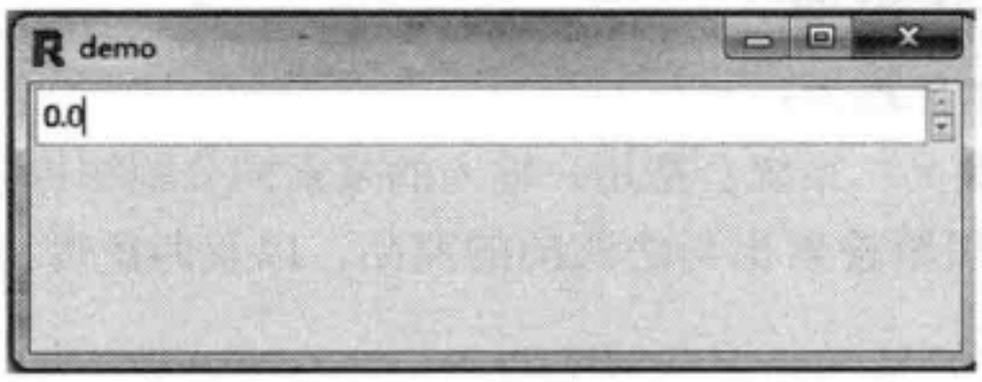


图 11-21 gspinbutton 控件示例

这里默认为最小值 0，如果输入数据大于 1，则 svalue 读取的数据为 1，反之，如果小于最小值 0，则读取为 0。单击右侧的向上和向下箭头后数值会自动加 0.1。

17. 图形展示控件 ggraphics

ggraphics 函数本身没有过多的参数，所以可以不定义而直接使用 add 函数将其添加到相关布局框中。代码如下：

```
win<-gwindow("demo");bg<-ggroup(cont=win, horizontal=FALSE)
add(bg, ggraphics())
```

以上实例只是显示了控件的内容，对于布局框和触发函数，可以参考 11.1.2 节引例中的说明，这里不再详细介绍了。

18. 控件的常用操作

控件可以支持的常用操作如表 11-3 所示。

表 11-3 控件的常用操作

操作说明	代码基本形式
读取控件数值	svalue(< 控件对象 >)
设置控件数值	svalue(< 控件对象 >)<-
控件 / 布局框是否可用	enabled(< 控件 / 布局框对象 >)<-T/F
控件 / 布局框是否可见	visible (< 控件 / 布局框对象 >)<-T/F
设置控件 / 布局框尺寸	size(< 控件 / 布局框对象 >)<-c(< 宽 >,< 高 >)
在布局框中添加某控件	add(< 布局框 >,< 控件 >)
在布局框中删除某控件	delete(< 布局框 >,< 控件 >)
消除某控件 / 布局框	dispose(< 控件 / 布局框对象 >)

11.2 实例：商品搜索算法参数配置监控 GUI 系统

电商类网站的商品搜索功能至关重要，网站的运营人员可以通过调节搜索算法内各权重值的大小来平衡不同等级的商品和店铺的曝光率。然而，很多情况下，不知如何确定合理的权重值。本例欲搭建一个 GUI 系统，以展示在设置不同的权重值时，各等级的商品和店铺的曝光率。

11.2.1 商品搜索算法概述

商品搜索系统的基本原理为：

- 分词并检索匹配商品：系统会把用户输入的搜索词分割解析为只含有最小语义的索引词，并使用搜索引擎检索出与之匹配的商品，以及匹配度得分，得分的取值范围是 (0,1]。
- 计算商品搜索得分：依据商品业务数据（包括商品所在店铺加降权、商品浏览量、商品收藏量、…、商品自身加降权）和各个指标的权重计算总体的得分情况。

商品搜索核心算法涉及的内容如图 11-22 所示。

商品搜索权重设置					
搜索权重项名称	权重范围	权重默认值	业务数据	比重分值，不可调整	单项权重计算（权重值*比重）
关键字匹配度	1~100	100	A=商品匹配度	6/15	权重默认值*商品匹配度*比重分值
店铺加降权	1~100	50	B=权重默认值设置，店铺默认值为50，单个店铺可以设置权重值	2/15	权重默认值*比重分值
商品浏览量	1~100	50	C=商品浏览数	1/15	权重默认值*tanh(商品浏览量/Max(商品浏览量))*比重分值
商品收藏	1~100	50	D=商品搜藏数	1/15	权重默认值*tanh(商品收藏数/Max(商品收藏数))*比重分值
销售量	1~100	50	E=商品销售数量	1/15	权重默认值*tanh(商品销售量/Max(商品销售数))*比重分值
商品评价得分	1~100	50	G=商品评分	1/15	权重默认值*tanh(商品评价/100)*比重分值
促销商品	1~100	50	H=促销标识0，1	1/15	权重默认值*tanh(0或1)*比重分值
商品加降权	1~100	50	I=权重默认值，商品默认值50，单个商品可以修改权重值	2/15	权重默认值*比重分值

图 11-22 商品搜索示例图

由图 11-22 可知，商品搜索的综合得分由 8 个指标组合而成，每个指标都有相应的权重值和比重分值。比重分值 × 权重值实际为各个指标占综合得分的比例，其中，比重分值实

现比例的重大调整，权重值则实现微调：每个商品和店铺的加降权指标都各自有一个权重值，运营人员可以提高或降低该值，以实现推广该商品或店铺的目的。其余指标的权重值则不区分商品和店铺，运营人员可以设置该类值实现不同等级的商品和店铺的整体曝光率。

此外，各指标的比重分值也不区分商品和店铺，“关键字匹配”至“商品加降权”等各指标的比例依次是 6:2:1:1:1:1:1:2，我们要求其和为 1，所以这里依次设置为 6/15、2/15、1/15、1/15、1/15、1/15、1/15、2/15，该值的调整对搜索系统影响极大，一般不调整。

因为某个指标的差异值很大（如不同商品的浏览量差异很大）时，可能会导致综合指标突变，所以应使用 tanh 双曲正切线函数来弱化该效应。$y=\tanh(x)$ 曲线如图 11-23 所示。

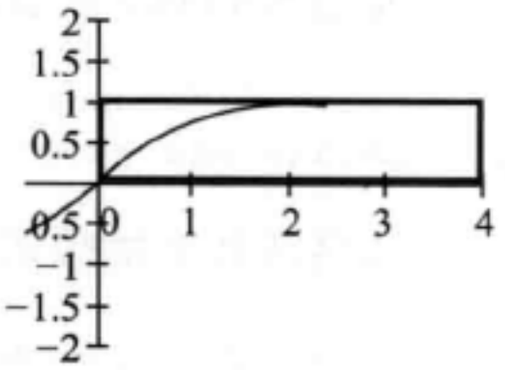


图 11-23 tanh 函数曲线

当自变量 x 的取值范围为 $[0,+\infty)$ 时，y 的取值范围为 [0,1]，即取值较大的 x 对 y 的影响被弱化了。

11.2.2 原始商品信息

原始数据存储于 data.csv 文件中（在实际应用中多以数据框形式存储，以实时更新数据，这里为了简化问题暂时以 csv 文件代替说明），其格式如图 11-24 所示（为 Excel 的部分截图）。

N6

	A	B	C	D	E	F	G	H	I	J
1		X	id	price	shop.id	like	review.score	buy.num	view.num	saleoff
2	1	1	1112228112	198	7152	1	50	0	100	FALSE
3	2	2	1096557612	398	7152	11	100	7	449	TRUE
4	3	3	1076239806	419	10869	30	88.9	9	608	FALSE
5	4	4	1211836402	199	5206	42	89.7	29	2061	FALSE
6	5	5	1078263406	129	10869	35	83.3	6	431	FALSE
7	6	6	1283163402	368	5206	21	50	4	268	FALSE
8	7	7	1008982822	118	4866	148	86.6	97	6992	FALSE
9	8	8	1182648508	168	9051	31	40	5	347	FALSE
10	10	10	1024458712	89	5372	452	95	301	21930	TRUE
11	13	13	1094419921	790	7127	56	71.4	7	498	FALSE
12	14	14	1108182121	89	6959	23	88.5	26	1832	FALSE
13	15	15	1449082201	229	7127	273	96.3	163	11305	FALSE
14	16	16	1099831105	149	8104	10	100	2	141	FALSE
15	17	17	1359214302	68	7040	803	92.1	560	39790	FALSE
16	18	18	1159302801	158	5551	343	94.9	235	15801	FALSE

图 11-24 数据集示例

在图 11-24 中，每列的含义如表 11-4 所示。

表 11-4 数据集说明

列名称	含义
	第一列和第二列均是行信息，对分析无意义
id	商品 ID
price	商品价格
shop.id	店铺 ID
like	商品被收藏的次数
review.score	商品评论得分，最高分为 100。如果商品未被销售，则默认得分为 50

（续）

列名称	含义
buy.num	商品销售数
view.num	商品浏览数
saleoff	是否为促销品，为 TRUE 时是促销品，为 FALSE 时不是促销品

1. 读入数据

使用 read.csv 函数读取源数据文件 data.csv，并存储于数据框对象 data 中，代码如下：

```
root="D:/chap11/xml/"
data<-read.csv(file=paste(root,"data.csv",sep=""))
```

商品数和店铺数依次为：

```
item.num<-nrow(data)
shop.num<-length(unique(data$shop.id))
```

2. 商品和店铺等级分类

本例的目的是查看不同等级商品和店铺的曝光率，这里规定的商品和店铺等级如表 11-5 所示。

表 11-5　商品和店铺的等级

商品等级 data$item.levels		店铺等级 data$shop.levels	
1 级	销售额（数量 x 单价）<= 500 元	1 级	销售额排名处于最末的 30 个店铺
2 级	500 元 < 销售额 <= 5 万元	2 级	中间的其他店铺
3 级	5 万元 < 销售额	3 级	销售额排名最前的 10 个店铺

添加商品等级（data$item.levels）的代码如下：

```
# 更新商品等级: data$item.levels
data<-cbind(data,sales.volume=data$buy.num*data$price)# 销量
data<-cbind(data,item.levels=1)
data$item.levels[which(data$sales.volume>500)]<-2# 销量大于 500 的商品等级为 2
data$item.levels[which(data$sales.volume>50000)]<-3# 销量大于 5 万的商品等级为 3
```

添加店铺等级（data$shop.levels）的代码如下：

```
# 更新店铺等级（依据店铺销量 = 销售商品数目 * 商品价格）:shop.levels，名字是商铺名称
shop.id<-unique(data$shop.id)
shop.sales<-c()
for(i in 1:shop.num){# 遍历所有店铺
    tmp.data<-data[which(data$shop.id==shop.id[i]),]
    shop.sales<-c(shop.sales,sum(tmp.data$price*tmp.data$buy.num))# 店铺的销量
}
names(shop.sales)<-shop.id
shop.levels<-sort(shop.sales)
shop.levels[1:30]<-1# 销量最后 30 是尾部商户
shop.levels[(length(shop.levels)-9):length(shop.levels)]<-3# 销量前 10 是尾部商户
```

```
shop.levels[31:(length(shop.levels)-10)]<-2#names 是 shop.id，值是等级
data<-cbind(data,shop.levels=shop.levels[as.character(data$shop.id)])
```

计算商品的销量额（data$sales.volume），并以此划分出商品等级，然后依次遍历店铺 ID，对店铺内各商品销售额求和（shop.sales），并依此来划分店铺等级。

以上代码属于商品数据的初始化更新，放在工作目录下的 init.data.txt 文件中。

11.2.3 系统概述

工作路径下包含的各文件如图 11-25 所示。



图 11-25 系统包含的文件

其文件内容如表 11-6 所示。

表 11-6 文件说明

文件名称	内容概述
data.csv	原始数据文件，参见 11.2.2 节所讲的内容
init.data.txt	用于读取原始数据文件的命令行文件（参见 11.2.2 节所讲的内容），最终得到数据框 data
init.fun.txt	用于更新各搜索指标的权重和比重分值，并运行 init.score.txt 文件来计算除关键字匹配得分以外的所有指标得分，返回数据框 score.1
init.score.txt	计算所有商品中除关键字匹配得分以外的指标得分
lib.txt	自定义函数的文件，包括： 1）gender.item() 函数：模拟用户的一次搜索行为，输入参数 num 为匹配出的商品数。输出一个 num 行 x2 列的数据框，其中 item.id 列为匹配的商品 ID，keywords.match.score 列为对应商品的关键字匹配度，范围是 (0,1] 2）reach.score() 函数：使用由 gender.item() 函数得出的对象来模拟本次搜索行为匹配出的商品 ID 和对应的搜索综合得分 total.score 3）display.score() 函数：批量模拟 search.num 次的搜索行为，并返回一个包含 3 列的数据框，其中 id 列是匹配出的并且符合曝光范围的商品 ID；item.levels 是该商品的等级，shop.levels 是该商品所在店铺的等级 4）plot.display.score() 函数：使用 display.score() 函数输出的数据框作为输入对象，并分析 item.levels 和 shop.levels 列的频数，从而得到不同等级商品和店铺的曝光率，最后绘制比例图 5）h.plot.display.score 把 plot.display.score() 函数封装成用于展示图像的 handler 触发函数
init.display.score.txt	通过调用 display.score() 函数来计算商品和店铺的曝光度数据 display.score.data
main.txt	建立 GUI 系统的主程序，包括相关控件的建立、布局和初始化

11.2.4 模块布局

11.1 节已经给出了 gui 系统中各个控件的布局，下面给出其中的关键代码，详尽代码存储于 main.txt 命令文件中。最终实现的布局如图 11-26 所示。

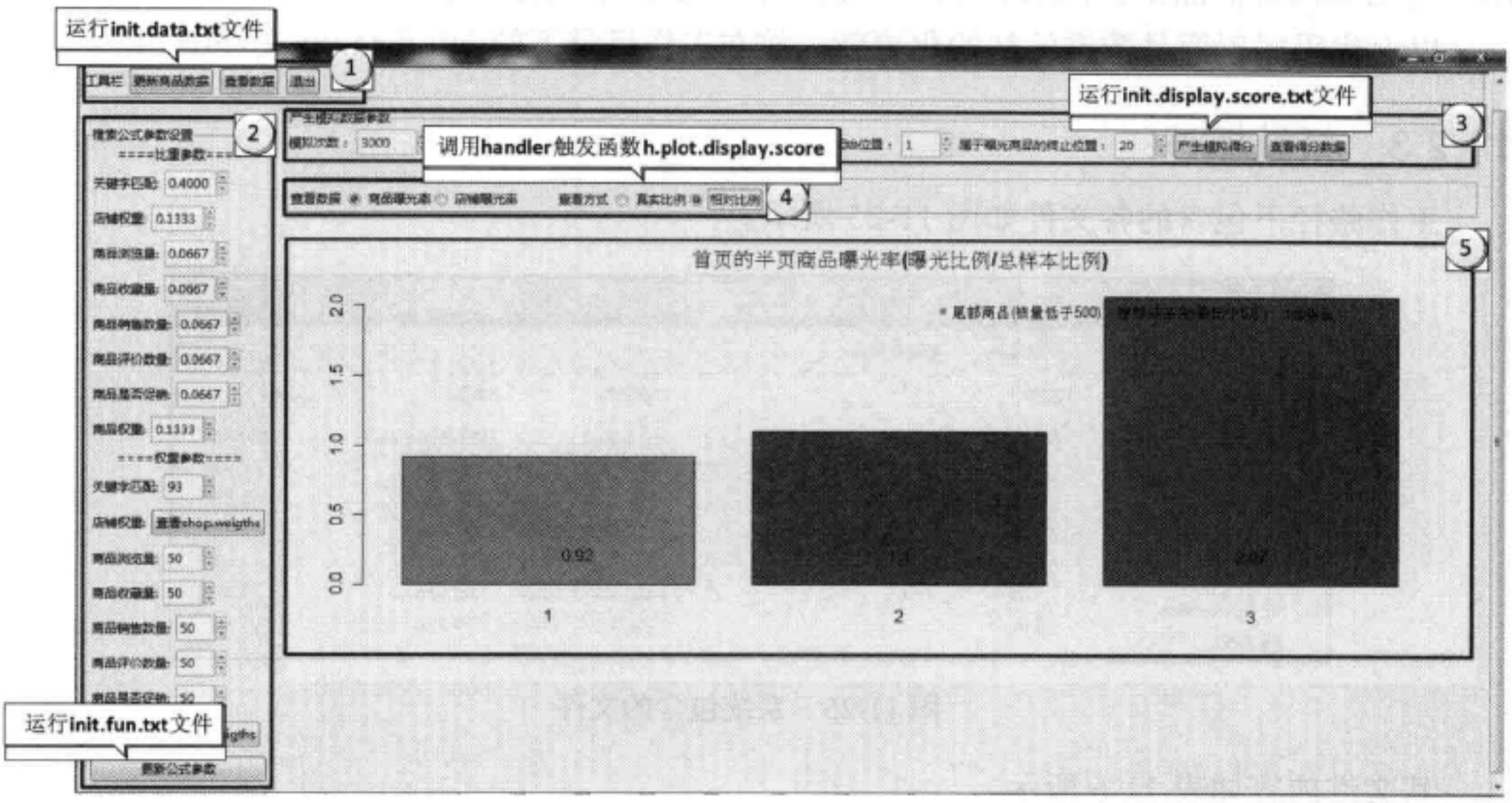


图 11-26 布局示例图

1. 建立窗口

定义一个标题为“搜索系统”的窗口，代码如下：

```
win<-gwindow("搜索系统")
```

2. 建立并布局控件，设置控件中的触发函数

1）在窗口对象 win 的基础上建立一个最低级的 ggroup 对象 b.gp，其中高度设置为 1200（参数 height 设置为 1200），并添加滚动条（参数 use.scrollwindow 设置为 T）。代码如下：

```
b.gp<-ggroup(cont=win,horizontal=F,height=1200,use.scrollwindow=T)
```

2）在 ggroup 对象 b.gp 的顶部添加“更新数据区域”的控件，代码如下：

```
tool.gf<-gframe(cont=b.gp)
glabel("工具栏",cont=tool.gf)
updata.button<-gbutton("更新商品数据",cont=tool.gf,handler=function(h,...)
source(paste(root,"init.data.txt",sep="")))
view.data.button<-gbutton("查看数据",cont=tool.gf,handler=function(h,...)
gdf(data,cont=T))
quit.button<-gbutton("退出",cont=tool.gf,handler=function(h,...) dispose(win))
```

主要的控件是“更新商品数据”按钮，单击该按钮后运行 init.data.txt 命令文件；“查

看数据”按钮，单击该按钮后使用 gdf 函数展示数据框 data。最后单击“退出”按钮调用 dispose(win) 命令来完成窗口的关闭动作。

3）在界面的左侧添加“设置搜索指标的权重和比重分值参数区域”控件，代码如下：

```
work.gp<-ggroup(cont=b.gp)
#### 左侧工具栏 ####
left.gf<-gframe(cont=work.gp,horizontal=F)
    parameter.gf<-gframe("搜索公式参数设置",cont=left.gf,horizontal=F)
    glabel("==== 比重参数 ====",cont=parameter.gf)
    keywords.proportion.gp<-ggroup(cont=parameter.gf)
    glabel("关键字匹配：",cont=keywords.proportion.gp)
    keywords.proportion.gsb<-gspinbutton(from=0,to = 1,by = 0.0001,value=2/5,
cont=keywords.proportion.gp)
    shop.proportion.gp<-ggroup(cont=parameter.gf)
        …< 其他输入控件 >…
  up.parameter.button<-gbutton("更新公式参数",cont=parameter.gf,handler=function (h,
...) source(paste(root,"init.fun.txt",sep="")))
```

对于权重和比重分值参数的设置主要使用 gspinbutton 控件。例如，在上面代码中，用于设置关键字匹配的比重参数的控件是：范围为 0 ～ 1，步进单位为 0.0001，默认值为 2/5 的 gspinbutton 控件。其余参数的设置控件也基本相同。

因为店铺和商品加降权的权重参数用于区分不同商品和店铺，所以不好在 GUI 系统直接设置。这里只给出查看按钮，用户只能通过控制台对对象 shop.weigths 和 item.weigths 进行赋值，从而完成设置。

“更新公式参数”按钮会调用运行 init.fun.txt 文件（内部会调用运行 init.score.txt 文件），并产生各商品中除关键字匹配得分以外的所有指标的得分 score.1。

4）在界面的右侧上方添加“产生模拟数据区域”控件，代码如下：

```
right.gg<-ggroup(cont=work.gp,expand=TRUE,horizontal=F)
    gen.data.parameter.gf<-gframe("产生模拟数据参数",cont=right.gg)
    glabel("模拟次数：",cont=gen.data.parameter.gf)
    search.num.gsb<-gspinbutton(from=1,to=1000000,by =1,value=50,
                    cont=gen.data.parameter.gf)
        …< 其他输入控件 >…
gen.data.button<-gbutton("产生模拟得分",cont=gen.data.parameter.gf,
      handler=function(h,...) source(paste(root,"init.display.score.
      txt",sep="")))
view.data.button<-gbutton("查看得分数据",cont=gen.data.parameter.gf,
      handler=function(h,...) gdf(display.score.data,cont=T))
```

首先使用 gspinbutton 控件设置 display.score 函数的 4 个相关参数。建立“产生模拟得分”按钮来调用运行 init.display.score.txt 文件（主要是调用 display.score 函数），模拟实现多次搜索行为，并计算曝光率数据 display.score.data。“查看得分数据”按钮会调用 gdf 函数来展示计算得到的 display.score.data 对象。

5）在界面右侧的中间添加“展示区的参数设置区域”控件，代码如下：

```
plot.gg<-ggroup(cont=right.gg,horizontal=F)
    plot.parameter.gf<-gframe(cont=plot.gg)
    glabel("查看数据",cont=plot.parameter.gf)
    is.item.value= c("商品曝光率","店铺曝光率")
    is.item.gd=gradio(is.item.value,cont=plot.parameter.gf,handler=h.plot.
                      display.score,horizontal=T)
    glabel("          查看方式",cont=plot.parameter.gf)
    is.real.value= c("真实比例","相对比例")
    is.real.gd=gradio(is.real.value,selected=2,cont=plot.parameter.gf,
                      handler=h.plot.display.score,horizontal=T)
```

这里主要是通过 gradio 控件来设置 plot.display.score 函数的参数，改变控件值时，会自动调用 plot.display.score 函数刷新下方图形区域里的内容。

6）添加“图形展示区域”控件，代码如下：

```
add(plot.gg,ggraphics())
```

3. 初始化

首次打开 GUI 系统时，初始化如下信息：

```
source(paste(root,"init.data.txt",sep=""))
source(paste(root,"init.fun.txt",sep=""))
source(paste(root,"init.display.score.txt",sep=""))
h.plot.display.score()
```

11.2.5 计算除关键字匹配以外的指标得分

计算除关键字匹配以外的指标得分的步骤如下：

1）读取用户在界面中通过控件输入的系统参数（代码存储于 init.fun.txt 文件中），以便于后续代码计算指标得分。代码如下：

```
# 部分指标的最大值
max.view.num<-max(data$view.num)# 最大浏览量
max.like<-max(data$like)# 最大收藏量
max.buy.num<-max(data$buy.num)# 最大购买量
max.review.score<-100# 最大商品评价得分
# 各指标的比重分值参数
proportion<-c(keywords=svalue(keywords.proportion.gsb),shop=svalue(shop.
              proportion.gsb),
              view.num=svalue(view.num.proportion.gsb),like=svalue(like.
              proportion.gsb),
              buy.num=svalue(buy.num.proportion.gsb),
              review.score=svalue(review.score.proportion.gsb),
              saleoff=svalue(saleoff.proportion.gsb),item=svalue(item.
              proportion.gsb))
# 各指标的权重参数
keywords.weigths<-svalue(keywords.weigths.gsb)
shop.weigths<-rep(50,shop.num);names(shop.weigths)<-unique(data$shop.id)
view.num.weigths<-svalue(view.num.weigths.gsb)
```

```
like.weigths<-svalue(like.weigths.gsb)
buy.num.weigths<-svalue(buy.num.weigths.gsb)
review.score.weigths<-svalue(review.score.weigths.gsb)
saleoff.weigths<-svalue(saleoff.weigths.gsb)
item.weigths<-rep(50,item.num);names(item.weigths)<-data$id
source(paste(root,"init.score.txt",sep=""))
```

上述代码首先使用 svalue 函数来读取“设置搜索指标的权重和比重分值参数区域”中的 gspinbutton 控件内容，然后调用运行 init.score.txt 文件来计算相关的指标得分。

2）计算除关键字匹配外的其他指标得分（代码存储于 init.score.txt 文件中）。代码如下：

```
score.1<-data.frame()# 用于存储所有商品中除了关键字匹配以外的所有指标得分
for(i in 1:item.num){# 遍历所有的商品
# 店铺得分 = 权重默认值 * 比重分值
shop.score<-shop.weigths[as.character(data[i,]$shop.id)]*proportion["shop"]
# 浏览量得分 = 权重默认值 *tanh( 商品浏览量 /Max( 商品浏览量 ))* 比重分值
view.num.score<-view.num.weigths*tanh(data[i,]$view.num/max.view.num)*
                proportion["view.num"]
# 收藏量得分 = 权重默认值 *tanh( 商品收藏数 /Max( 商品收藏数 ))* 比重分值
like.score<-like.weigths*tanh(data[i,]$like/max.like)*proportion["like"]
# 销量得分 = 权重默认值 *tanh( 商品销售量 /Max( 商品销售数 ))* 比重分值
buy.num.score<-buy.num.weigths*tanh(data[i,]$buy.num/max.buy.num)
                *proportion["buy.num"]
# 评价得分 = 权重默认值 *tanh( 商品评价 /100)* 比重分值 ,100 为最大允许的得分
review.score<-review.score.weigths*tanh(data[i,]$review.score/max.review.score)
                *proportion["review.score"]
# 促销得分 = 权重默认值 *tanh(0 或 1)* 比重分值
saleoff.score<-saleoff.weigths*tanh(as.numeric(data[i,]$saleoff))
                *proportion["saleoff"]
# 商品加降权得分 = 权重默认值 * 比重分值
item.score<-item.weigths[as.character(data[i,]$id)]*proportion["item"]
score.1<-rbind(score.1,c(data[i,]$id,data[i,]$item.levels,data[i,]$shop.id,
                data[i,]$shop.levels,shop.score,view.num.score,like.score,
                buy.num.score,review.score,saleoff.score,item.score))
}
names(score.1)<-c("id","item.levels","shop.id","shop.levels","shop.score",
                  "view.num.score","like.score","buy.num.score","review.score",
                  "saleoff.score","item.score")
```

上面的代码主要是依据搜索得分的计算公式来进行计算，最后得到 score.1 数据对象。

11.2.6 模拟一次搜索行为数据

11.2.5 节只得到了所有商品中除关键字匹配得分以外的指标得分，并不能得到最后的综合得分。下面通过函数 gender.item 和 reach.score 来模拟一次搜索行为产生的匹配商品 ID 和匹配度得分，并计算商品的综合得分 total.score。

1. gender.item 函数

自定义 gender.item 函数（代码存储于 lib.txt 文件中）的输入参数 num，用于设置本次搜

索到的匹配的商品数。如果 num 大于 data 的商品数 item.num，则提示输入参数 num 有误。

```
gender.item<-function(num){# 模拟搜索的字符匹配
    if(num>item.num){
        print("num 输入错误！！ ");return()
    }
    keywords.score<-rnorm(num,0.6,0.2)
    item.id<-data[sample(1:item.num,num),]$id
    result<-data.frame(item.id=item.id,keywords.match.score=keywords.score)
}
```

假设每个商品的匹配概率相同，则使用 sample 函数对 data 进行无放回的抽样，从而得到匹配的商品编号 item.id。代码如下：

```
item.id<-data[sample(1:item.num,num),]$id
```

通过对历史数据的研究，我们认为每次搜索的商品匹配得分服从均值为 0.6，标准差为 0.2 的正态分布（对于商品匹配的分布模型这里不做深入研究，读者可以使用大量的历史数据并通过 nls 函数来拟合处分布模型曲线）。代码如下：

```
keywords.score<-rnorm(num,0.6,0.2)
```

最后把抽样出的商品编号 item.id 和对应的关键字匹配度 keywords.score 组合为数据框输出。

2. 函数 reach.score

自定义 reach.score 函数（代码存储于 lib.txt 文件中）的输入参数 reach.id 和 keywords.match.score 指的是搜索出的商品 ID 和关键字匹配度，由上文的 gender.item 函数产生。这两个参数涉及的代码如下：

```
reach.score<-function(reach.id,keywords.match.score){
    reach.num<-length(reach.id)# 搜索出的商品个数
    result<-data.frame()
    for(i in 1:reach.num){
        index<-which(score.1$id==reach.id[i])# 搜索出的商品所在 socre.1 的行序列号
        # 关键字得分 = 权重默认值 * 商品匹配度 * 比重分值
        keywords.score<-keywords.weigths*keywords.match.score[i]*proportion
        ["keywords"]
        total.score<-keywords.score+sum(score.1[index,c("shop.score",
                    "view.num.score","like.score","buy.num.score","review.
                    score",
                    "saleoff.score","item.score")])# 搜索的综合得分
        result<-rbind(result,c(score.1[index,],keywords.score,total.score))
    }
    names(result)[12:13]<-c("keywords.score","total.score")
    result
}
```

首先遍历搜索的匹配商品 ID，并计算该商品的关键字指标得分。代码如下：

```
# 关键字得分 = 权重默认值 * 商品匹配度 * 比重分值
keywords.score<-keywords.weigths*keywords.match.score[i]*proportion["keywords"]
```

然后计算所有指标的综合得分 total.score。代码如下：

```
total.score<-keywords.score+sum(score.1[index,c("shop.score",
            "view.num.score","like.score","buy.num.score","review.score",
            "saleoff.score","item.score")])# 搜索的综合得分
```

11.2.7 模拟多次搜索行为数据

使用 display.score 函数批量模拟 search.num 次的搜索行为数据，并计算出不同等级的商品和店铺曝光度。相关代码存储于 lib.txt 命令文件中。

```
display.score<-function(search.num,match.items.num,think.num.min,think.num.max){
    result<-data.frame()
    for(i in 1:search.num){
        tmp<-gender.item(match.items.num[i])
        reach.score.tmp<-reach.score(tmp$item.id,tmp$keywords.match.score)
        reach.score.dec<-reach.score.tmp[order(reach.score.tmp$total.score,
                        decreasing=T),]# 依据总分排序
        # 只选择指定范围内的曝光商品 id 号和商品等级以及商铺等级
        result<-rbind(result,reach.score.dec[think.num.min:think.num.max,
                c("id","item.levels","shop.levels")])
    }
    result
}
```

其中，输入参数 search.num 是模拟搜索行为的次数；match.items.num 参数是长度为 search.num 的数据向量，用于表示每次搜索行为的匹配商品数，即 gender.item() 函数中的输入参数 num ；think.num.min 和 think.num.max 参数用于表示欲统计的曝光商品的排名范围。例如，如果网站的商品搜索列表页是 10 行 4 列的排序布局，而想要研究首页上半页商品和店铺的曝光率，则可以设置 think.num.min=1，think.num.max=20。如果要研究第二页的曝光率，则设置 think.num.min=41，think.num.max=80。

函数最后返回一个包含 3 列的数据框，其中 id 列是匹配出的并且符合曝光范围的商品 ID，item.levels 是该商品的等级，shop.levels 是该商品所在店铺的等级。

11.2.8 绘图

绘图方法的实现，主要是指“展示区的参数设置区域”中 gradio 控件的 handler 触发函数 h.plot.display.score() 及其内部核心函数 plot.display.score() 的实现方案。相关代码存储于 lib.txt 命令文件中。

1. 核心函数 plot.display.score()

输入参数 display.score 是由 display.score() 函数输出的对象；如果 is.item 参数为 1，则

绘制商品曝光率；如果为 2，则绘制店铺曝光率；如果 is.real 参数为 1，则绘制实际曝光比例，为 2，则绘制曝光能力 = 曝光比例 / 总样本比例。代码如下：

```
plot.display.score<-function(display.score,is.item=1,is.real=1){
    par(family="GB1")
    color<-c("green","blue","red")
    if(is.item==1){# 绘制商品曝光率
        if(is.real==1){# 绘制真实比例
            plot.data<-table(display.score$item.levels);
            plot.data<-plot.data/sum(plot.data)
            barplot(plot.data,col=color[as.numeric(names(plot.data))],
                    main=" 商品曝光率（曝光比例）")
            legend("topright",legend=c(" 尾部商品（销量低于 500)"," 腰部商品（销量低于 5
            万）",
                    "top 商品 "),horiz=T,pch=15,col=color,cex=0.8,bty="n")
            text(c(0.8,1.9,3.1),0.1*max(plot.data),paste(round(plot.
            data*10000)/100,
                  "%",sep=""))
        }else{# 绘制相对比例
            plot.data1<-table(display.score$item.levels);
            plot.data1<-plot.data1/sum(plot.data1)
            plot.data2<-table(data$item.levels);
            plot.data2<-plot.data2/sum(plot.data2)
            plot.data<-plot.data1/plot.data2
            barplot(plot.data,col=color[as.numeric(names(plot.data))],
                    main=" 商品曝光率（曝光比例 / 总样本比例）")
            legend("topright",legend=c(" 尾部商品（销量低于 500)"," 腰部商品（销量低于 5
            万）",
                    "top 商品 "),horiz=T,pch=15,col=color,cex=0.8,bty="n")
            text(c(0.8,1.9,3.1),0.1*max(plot.data),round(plot.data*100)/100)
        }
    }else{
        ###### 绘制店铺曝光率部分省略 ######
        }
}
```

为了在输出图形区域显示汉字，首先设置 par(family="GB1")。如果 is.item==1 且 is.real==1，则要绘制不同等级商品的曝光率，由于 display.score$item.levels 代表曝光商品中的商品等级，因此要展示不同等级的商品曝光率，实际就是分析 display.score$item.levels 的比例。核心代码如下：

```
plot.data<-table(display.score$item.levels);
plot.data<-plot.data/sum(plot.data)
barplot(plot.data,col=color[as.numeric(names(plot.data))],main=" 商品曝光率（曝光比例）")
```

当模拟 3000 次搜索行为时，搜索列表页的首页上半页 (think.num.min=1，think.num.max=20) 的商品等级曝光结果如图 11-27 所示。

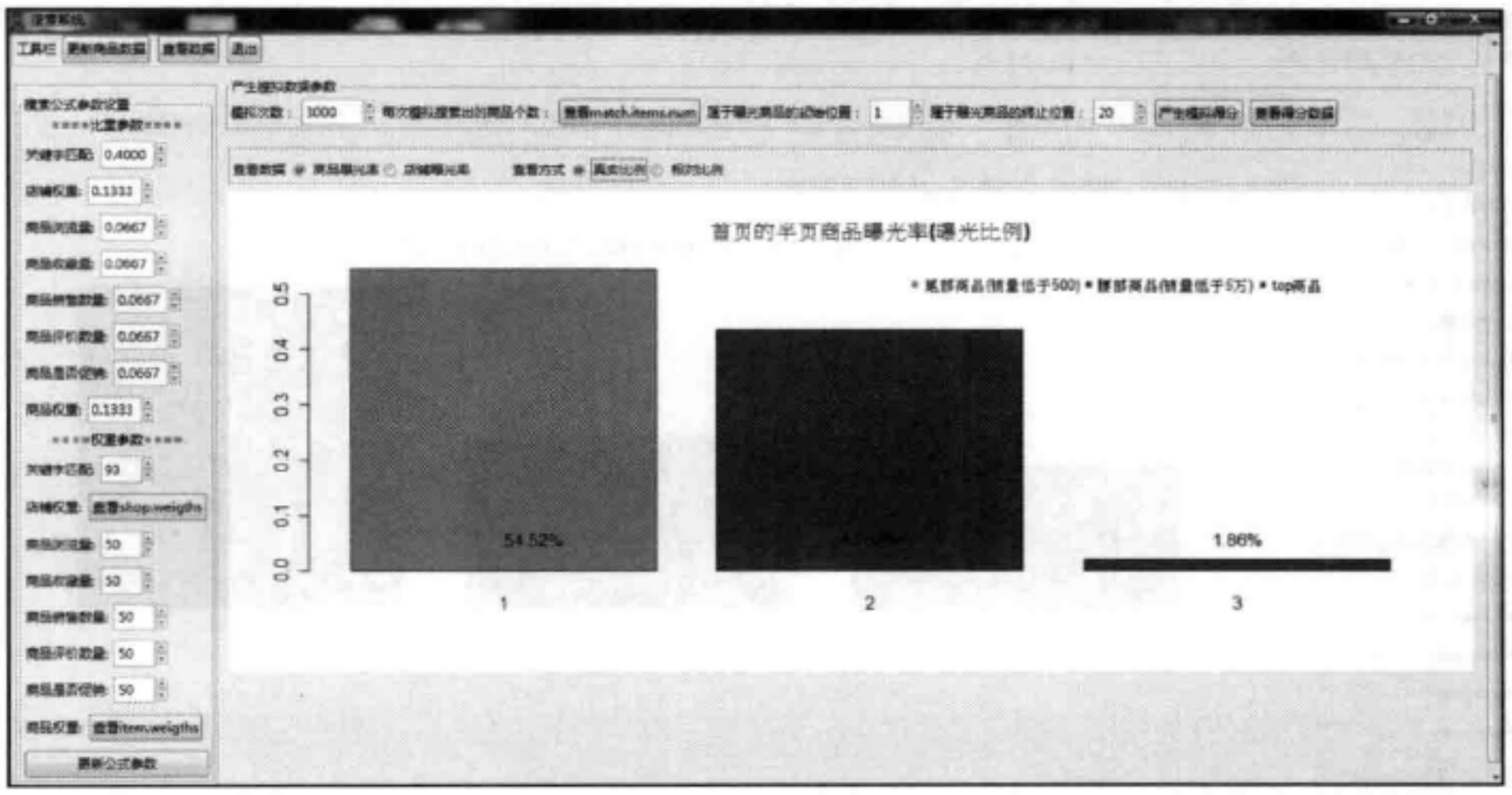


图 11-27 商品等级曝光结果

从图 11-27 中可以看到，尾部商品：腰部商品：top 商品 =54.52% : 43.62% : 1.86%。由于 top 商品（1 级商品）本身的样本比例很少（table(data$item.levels) 的 1 级商品比例），所以其曝光比例也不会过高，该指标并不能衡量不同等级商品的曝光能力。为了解决这个弊端，引入曝光能力的概念：

曝光能力 = 曝光比例 / 样本比例

如果 is.item==1 且 is.real!=1，则要绘制不同等级商品的曝光能力，其核心代码如下：

```
plot.data1<-table(display.score$item.levels);
plot.data1<-plot.data1/sum(plot.data1)
plot.data2<-table(data$item.levels);
plot.data2<-plot.data2/sum(plot.data2)
plot.data<-plot.data1/plot.data2
barplot(plot.data,col=color[as.numeric(names(plot.data))],
        main="商品曝光率（曝光比例 / 总样本比例）")
```

其中 plot.data1 是真实曝光比例，plot.data2 是总体样本中的不同等级比例。仍模拟 3000 次后的结果如图 11-28 所示。

可见 top 商品的曝光能力为 2.071>1，即推荐系统对 top 商品的曝光度很高；而腰部商品的曝光能力为 1.1，约为 1，即系统对其曝光略有提高；而尾部商品的曝光能力为 0.92，即系统对这部分商品的曝光度有所减少。

店铺部分的曝光能力与商品部分类似，这里就不展示代码了。

2. 触发函数 h.plot.display.score()

核心函数 plot.display.score() 并不能直接作为控件的触发函数，因其输入参数 is.item 和 is.real 也是通过 radio 控件输入的，所以这里再封装一层函数 h.plot.display.score()。代码如下：

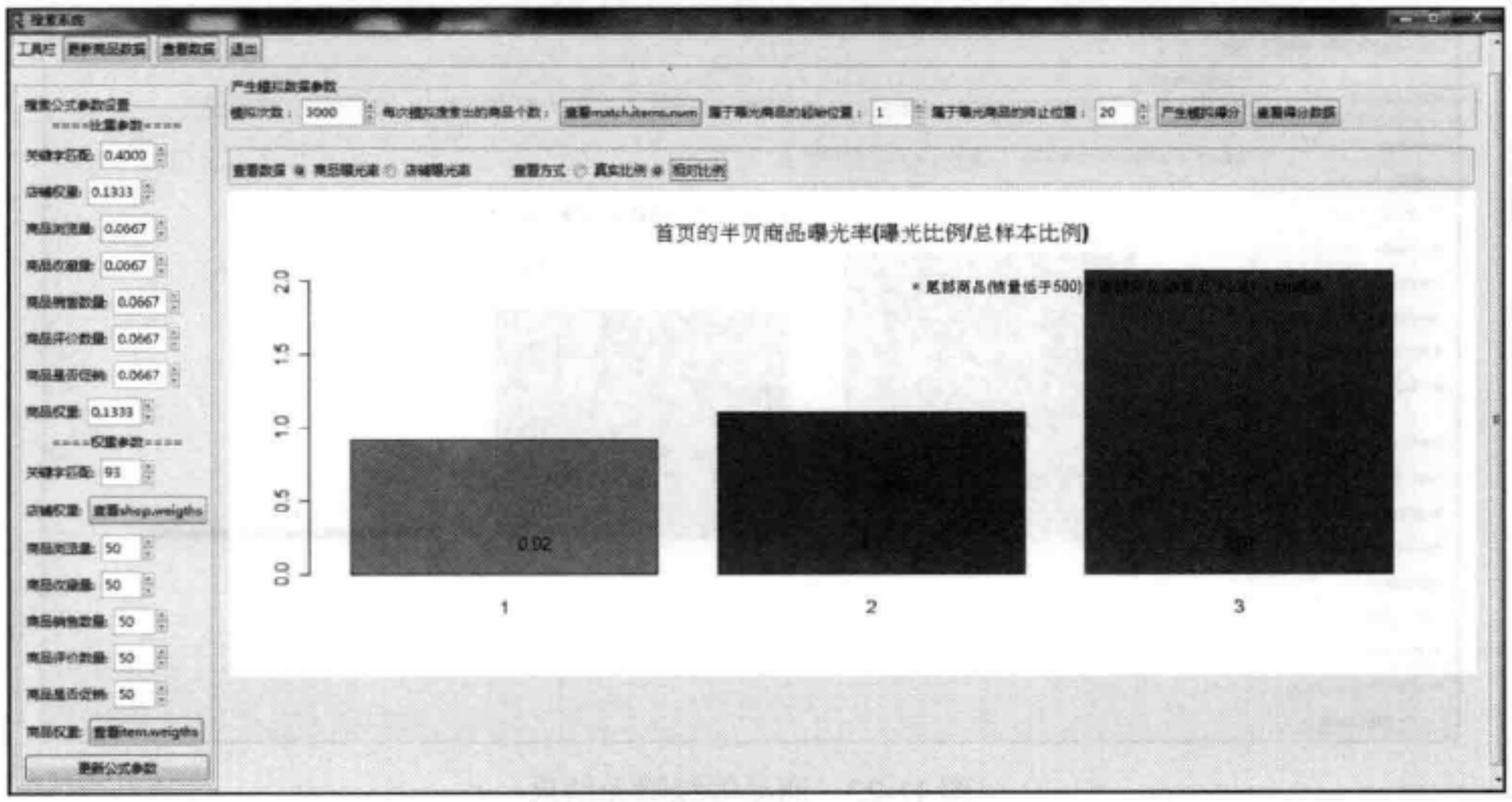


图 11-28 模拟 300 次后的结果

```
h.plot.display.score<-function(h,...){
    is.item<-which(is.item.value==iconv(svalue(is.item.gd),"UTF-8","gbk"))
    is.real<-which(is.real.value==iconv(svalue(is.real.gd),"UTF-8","gbk"))
    plot.display.score(display.score.data,is.item,is.real)
}
```

首先使用 svalue 函数接收控件的输入数据，然后调用核心函数 plot.display.score 完成图形展示。

第 12 章

基于 RFM 模型的客户价值系统

网站的运营人员通常会希望把浏览网站的用户划分为不同的等级。Wyner 于 1996 年指出，企业 80%的销售利润来自于 20%的顾客，而其余 20%的销售利润，却花了公司 80%的推广费用。由此可知，对用户建立一整套顾客价值分析体系，并针对不同价值等级的用户进行管理很重要。

Hughes 于 1994 年提出了 RFM 顾客价值分析模型，此模型利用 3 种指针：最近购买日（recency）、购买频率（frequency）及购买金额（monetary）来判断顾客的价值。该方法是目前企业最常用的顾客价值分析方法之一。

本章首先介绍马尔科夫链原理，并结合 RFM 模型了解客户价值的核心算法。

12.1 马尔科夫链

12.1.1 引例

下面是一组序列的定义式：

$$x_i = x_{i-1}/y_{i-1} + \text{随机误差}$$

$$y_i = y_{i-1}/x_{i-1} + \text{随机误差}$$

其中随机误差服从标准正态分布。代码如下：

```
x<-c();y<-c();
x[1]<-2;y[1]<-2
num<-15
for(i in 2:num){
    x[i]<-x[i-1]/y[i-1]+rnorm(1)
    y[i]<-y[i-1]/x[i-1]+rnorm(1)
}
```

试以 x 为横轴，y 为纵轴绘制出散点图，并使用射线连接点 $(x_{i-1}, y_{i-1}) \rightarrow (x_i, y_i)$。代码如下：

```
plot(x,y)# 绘制散点图
for(i in 2:num){# 绘制射线
    arrows(x[i-1],y[i-1],x[i],y[i])
}
```

最终显示的图形如图 12-1 所示。

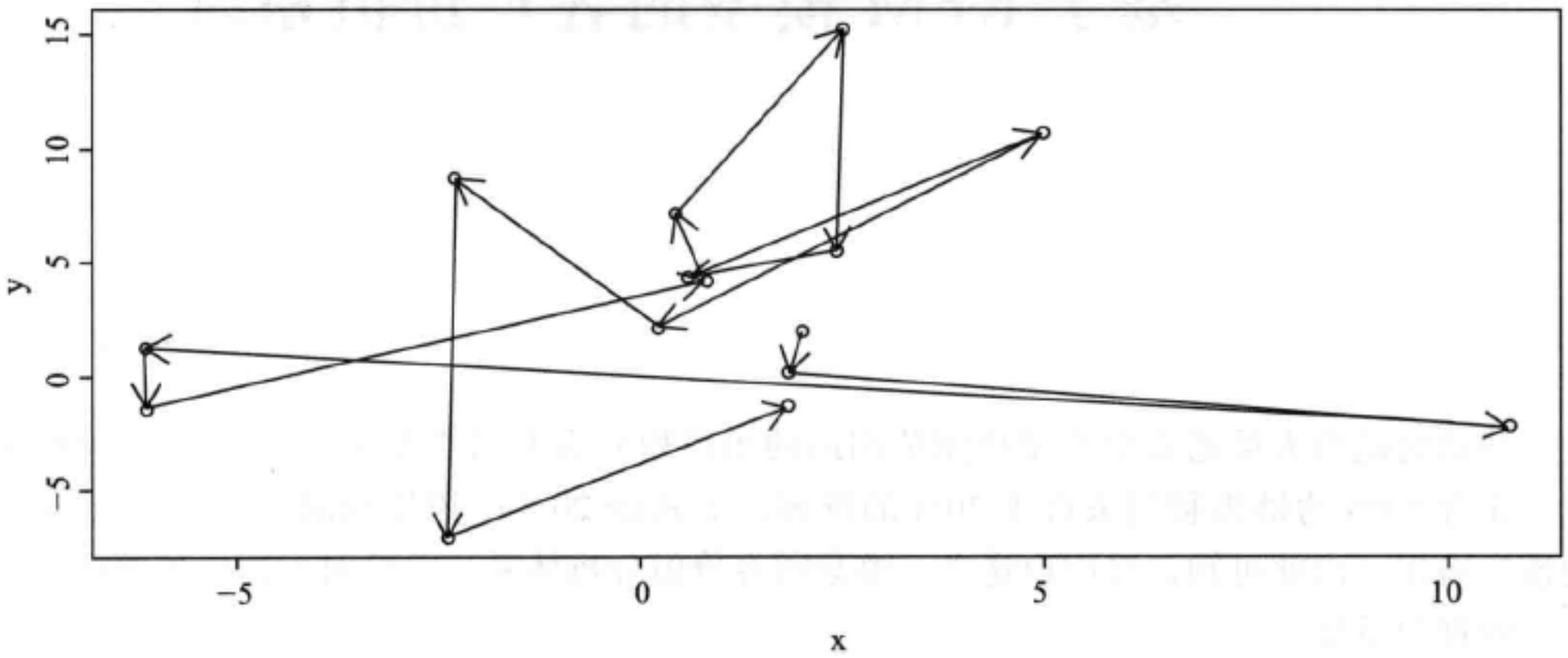


图 12-1　马尔科夫链示例

本引例有一个特性：本期数据 (x_i, y_i) 只依赖于上一期数据 (x_{i-1}, y_{i-1})，这就是一个马尔科夫链的实例。

12.1.2　概念

1. 马尔科夫链

在转移过程中，有一类事物在某种因素作用下，它们的状态概率取决于前一次的结果。例如，第 n 次结果的概率规律仅取决于第 $(n\text{-}1)$ 次试验的结果，第 $(n\text{-}1)$ 次试验结果仅取决于第 $(n\text{-}2)$ 次结果等，而与更早的结果无关，符合该类特性的叫马尔科夫链，其数学公式如下：

$$P(x_{n+1}=j \mid x_n=i_n, x_{n-1}=i_{n-1}, \cdots\cdots x_1=i_1)=P(x_{n+1}=j \mid x_n=i_n)$$

定义：$Pij = P(xt+1 = j \mid xt = i)$

2. 状态转移概率

状态转移概率的含义是：在本状态 x_n 等于 i 的条件下，使下一个状态 x_{n+1} 等于 j 的条件概率，即从 i 状态一步转移到 j 状态的概率，其数学公式如下：

$$Pij = P(x_{n+1}=j \mid x_n=i)$$

12.1.3　预测实例

表 12-1 是某电商网站 4 个用户在 2013 年上半年里每个月的消费金额。

表 12-1 数据集

	1 月份	2 月份	3 月份	4 月份	5 月份	6 月份
用户 1	0	129.9	131.5	137.8	0	125.4
用户 2	130	132	135.1	129.6	126	137
用户 3	135.8	135	134.9	0	137	0
用户 4	138.3	0	140.3	137.7	138.2	0

这组数据中的最大值为 140.3，最小值为 125.4，因此可以将取值范围划分为 [125.4,129.125],[129.125,132.85],[132.85,136.575],[136.575,140.3]，如果依次将这些范围设置为 A～D，未消费的用户设置为 E，那么，观测数据将划分为如表 12-2 所示的形式。

表 12-2 月消费状态

月消费状态	A	B	C	D	E
消费区间	[125.4,129.125]	[129.125,132.85]	[132.85,136.575]	[136.575,140.3]	0
频数	2	5	4	7	6

转化后的月消费数据集如表 12-3 所示。

表 12-3 转换后的月消费数据集

	1 月份	2 月份	3 月份	4 月份	5 月份	6 月份
用户 1	E	B	D	D	E	A
用户 2	B	B	C	B	A	D
用户 3	C	C	C	E	D	E
用户 4	D	E	D	D	D	E

从表 12-3 中可以看到，用户 1 的消费历史是：E → B; B → D; D → D; D → E; E → A，也就是说，他的状态转移情况如表 12-4 所示。

表 12-4 用户 1 的状态转移

本月 / 上个月	A	B	C	D	E
A	0	0	0	0	0
B	0	0	0	1	0
C	0	0	0	0	0
D	0	0	0	1	1
E	1	1	0	0	0

那么，表 12-3 中所有用户（4 个）的消费状态转移情况可以转化为表 12-5 所示的形式。

表 12-5 所有用户的状态转移

本月 / 上个月	A	B	C	D	E
A	0	0	0	1	0
B	1	1	1	1	0

（续）

本月 / 上个月	A	B	C	D	E
C	0	1	2	0	1
D	0	0	0	3	4
E	1	1	0	2	0

由此得到状态转移矩阵为：$P=\begin{bmatrix} 0 & 0 & 0 & 100\% & 0 \\ 25\% & 25\% & 25\% & 25\% & 0 \\ 0 & 25\% & 50\% & 0 & 25\% \\ 0 & 0 & 0 & 42.85\% & 57.15\% \\ 25\% & 25\% & 0 & 50\% & 0 \end{bmatrix}$

如果用户 5 在 6 月份的消费金额为 134.8 元，即该用户的月消费状态为 C。通过状态转移矩阵 P 可预测用户 5 在接下来半年的消费状态如表 12-6 所示。

表 12-6 预测消费状态

月份	L(n) 计算公式	L(n) 结果
6 月份	L(6)=[0,0,1,0,0]	[0,0,1,0,0]
7 月份	L(7)= L(6)*P	[0 ,0.25 ,0.5 ,0 ,0.25]
8 月份	L(8)= L(7)*P= L(6)*P^2	[0.125 ,0.25 ,0.3125 ,0.1875 ,0.125]
……		

使用下面代码计算 8 月份的消费状态 L(8)。

```
    p<- matrix(c(0,1/4,0,0,1/4,0,1/4,1/4,0,1/4,0,1/4,1/2,0,0,1,1/4,0,3/7,1/2,0,0,1/4,4
/7,0),5)
    l.6<-matrix(c(0,0,1,0,0),1)
    l.8<-l.6%*%p%*%p
```

然后根据表 12-2，使用消费区间的均值代表消费状态下的估值金额，如表 12-7 所示。

表 12-7 不同消费区间的估值金额

月消费状态	A	B	C	D	E
消费区间	[125.4,129.125]	[129.125,132.85]	[132.85,136.575]	[136.575,140.3]	0
估值金额	127.2625	130.9875	134.7125	138.4375	0

最后预测用户 5 在接下来半年时的消费金额如表 12-8 所示。

表 12-8 预测未来用户 5 的消费金额

月 份		
6 月份	M(6)=0*127.2625+ 0*130.9875+1* 134.7125+0* 138.4375+0*0	
7 月份	M(7)= 0*127.2625+ 0.25*130.9875+0.5* 134.7125+0* 138.4375+0.25*0	100.1031
8 月份	M(8)= 0.125*127.2625+ 0.25*130.9875+0.3125* 134.7125+0.1875* 138.4375+0.125*0	116.7094
……		

使用下面代码计算 8 月份的消费金额 M(8)。

```
l<-matrix(c(127.2625, 130.9875, 134.7125, 138.4375,0),5)
m.8<-l.8%*%l; m.8
          [,1]
[1,] 116.7094
```

12.2 RFM 模型

上节使用马尔科夫链建立月消费状态的转移矩阵，并使用它最终预测用户在未来月份的消费金额。对于一个电商网站，我们不能单纯地使用用户月消费金额来衡量用户的价值。一般可使用最近购买日（recency）、购买频率（frequency）及购买金额（monetary）这 3 个指标综合衡量用户价值，即 RFM 模型。

12.2.1 RFM 的概念

在 RFM 概念中，R 表示自最近发生购买行为之期计起，距现在已有几期没有发生购买行为。

F 表示顾客最近一期发生购买行为时的购买频率（当期内购买了几次）。

M 表示顾客最近一期发生购买行为时的平均单次购买金额（由于购买金额为连续变量不易于使用马尔科夫链分析，因此常常把它转化为离散变量：购买状态，这里与 12.1.3 节的实例类似）。

举例说明，如果用户在 2013.2.24 购买了 2 次商品，其总金额为 1009 元，在 2013.2.28 购买了 1 次商品，其金额为 547 元，则 2013.2.24—2013.3.2 内（以天为周期）的 RFM 数据如表 12-9 所示。

表 12-9 RFM 示例

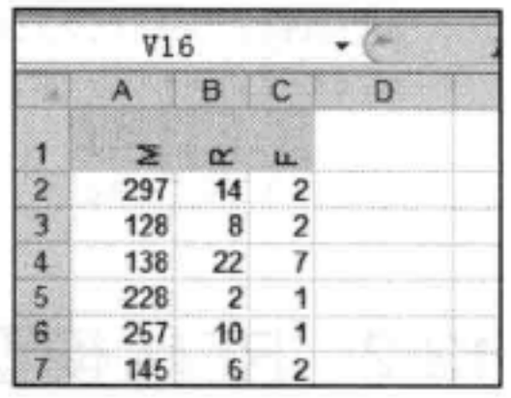
日期	2.24	2.25	2.26	2.27	2.28	3.1	3.2
当日是否发生购买行为	是	否	否	否	是	否	否
R	0	1	2	3	0	1	2
当日的购买次数	2	0	0	0	1	0	0
F	2	2	2	2	1	1	1
当日的购买金额	1009	0	0	0	547	0	0
当日的平均单次购买金额	1009/2=504.5	0	0	0	547/1=273.5	0	0
M	504.5	504.5	504.5	504.5	273.5	273.5	273.5

在建模过程中，只需使用某一个周期的 RFM 数据（如 2013.3.2 这一天的数据），并以此为基础计算后面的状态转移矩阵，最后预测用户在 2013.3.2 以后的购买状态。

12.2.2 数据集

下面的数据集中包含了某网站 2012.2.6—2012.2.19 内的 RFM 数据（RFM 统计方式是以

2 个星期为周期），其中 F 可以使用周期内的用户订单数来表示，M 可以使用周期内的用户平均订单金额来表示。共 4000 条数据。原始数据集存储于 data.xls 文件内，使用 Excel 打开后的部分截图如图 12-2 所示。



	A	B	C	D
1				
2	297	14	2	
3	128	8	2	
4	138	22	7	
5	228	2	1	
6	257	10	1	
7	145	6	2	

图 12-2 数据集部分截图

注意，这里的数据集是经过 RFM 概念处理过的，而非单纯的是统计期间内所在周期的数据。

使用如下代码读取数据集：

```
root<-"E:/chap12/"
file<-paste(root,"data.xls",sep="")
library (RODBC)
excel_file <- odbcConnectExcel(file)
data<- sqlFetch ( excel_file,"data")[,1:3]
close ( excel_file )
```

读入后，RFM 数据存放于数据框 data 中，其列名称依次为 R、F、M。

12.2.3 购买行为随机模型

1. 前提假设

在建立购买行为随机模型之前，先做 6 个假设，并利用此 6 个假设描述顾客购买行为，从而构建相关的随机模型。本随机模型结合了 Ehrenberg 提出的负二项分配模型及 Colombo & Jiang 提出的 Gamma-Gamma 混合型模型，该模型设立顾客每期购买频率及购买金额的机率分配假设，并利用 Gamma 分配捕捉顾客的异质性，从而描述顾客的购买行为。

假设一：假设顾客购买频率和购买金额这两个不同的行为维度是互相独立，不具有相关性。因此这两个行为机率函数的参数互相独立。

假设二：假设顾客的购买状态转移行为符合马可夫链的假设，则表示顾客下一期购买状态发生的机率只和上一期的购买状态有关。

假设三：假设个别顾客购买频率 f 为卜松分配（poisson distribution）：

$$P_f[F=f|\lambda]=e^{-\lambda}\frac{\lambda^f}{f!}\ \lambda>0 \tag{12-1}$$

公式 12-1 表示，在单位时间平均购买次数为 λ 的情况下，单位时间内购买次数为 f 的机率。

假设四：因为考虑顾客的异质性，故假设个别顾客单位时间平均购买次数 λ 服从 Gamma 分配：

$$g_\lambda(\lambda|w,\alpha)=\frac{\alpha^w}{\Gamma(w)}\lambda^{w-1}e^{-\alpha\lambda}\ w>0,\ \alpha>0 \tag{12-2}$$

其中 $\Gamma(w)$ 代表 gamma 函数。

假设五：假设个别顾客发生购买行为之各期平均单次购买金额服从 Gamma 分配，因为购买金额不可能为负，不适合用常态分配来捕捉，因此依据 Colombo & Jiang 的假设，采用更具有弹性、并且符合购买金额不为负之特性的 Gamma 分配：

$$g_m(m|u,\theta)=\frac{\theta^u}{\Gamma(u)}m^{u-1}e^{-\theta m} \quad u>0,\ \theta>0 \tag{12-3}$$

在公式 12-3 中，m 代表各期平均单次购买金额。

假设六：依据 Colombo & Jiang 的假设，由于顾客各期平均单次购买金额服从 Gamma 分配的平均值为 θ/u，为了考虑顾客的异质性，假设此 Gamma 分配的平均值 θ/u 随不同顾客而变动，因此，将 u 定义为常数值，利用 θ 捕捉每位顾客购买金额行为的不同，假设顾客平均单次购买金额的 Gamma 分配之参数 θ 符合另一个 Gamma 分配：

$$g_\theta(\theta|v,\phi)=\frac{\phi^v}{\Gamma(v)}\theta^{v-1}e^{-\phi\theta} \quad v>0,\ \phi>0 \tag{12-4}$$

2. 购买频率的概率

根据假设三和假设四可以推导出顾客购买频率的机率为负二项分配（Ehrenberg 于 1959 年提出）：

$$P_{NBD}[\mathrm{F}=f|\alpha,w]=\frac{\Gamma(w+f)}{\Gamma(w)f!}\left(\frac{\alpha}{\alpha+1}\right)^w\left(\frac{1}{\alpha+1}\right)^f \tag{12-5}$$

使用公式 12-5，可以计算用户的购买频率等于 f 时的概率。其中函数 gamma() 用来实现伽玛函数。函数 factorial() 是 n 阶乘函数，例如，factorial(5) 等于 $1\times2\times3\times4\times5=5!$。公式 12-5 的 R 语言代码如下：

```
prob.f<-function(f,a,w){
    result<-gamma(w+f)/(gamma(w)*factorial(f))
    result<-result*(a/(a+1))^w
    result<-result*(1/(a+1))^f
    result
}
```

如果设置 a=1.5，w=2.5，则绘制 f 值为 1:20 的概率直方图的代码如下：

```
a<-1.5;w<-2.5
f<-c()
for(i in 1:20){
    f[i]<-prob.f(i,a,w)
}
barplot(f)
```

绘制结果如图 12-3 所示。

每条直方图的纵轴代表相应 f 值下的概率。

利用 bayes 理论可以预测用户 i 在下一周期的购买次数 f_i 的期望值 $E(f_i)$，其公式如下：

$$E(f_i)=\frac{w+x_i}{\alpha+1}$$

其中，f_i 是第 i 个顾客的期望购买频率（购买次数），w 和 α 是公式 12-5 中的参数，x_i 是指顾客在上一个周期内购买的次数。可以通过顾客过去的购买次数预测顾客未来的购买次

数，这就是预测顾客未来购买次数的模型基础。

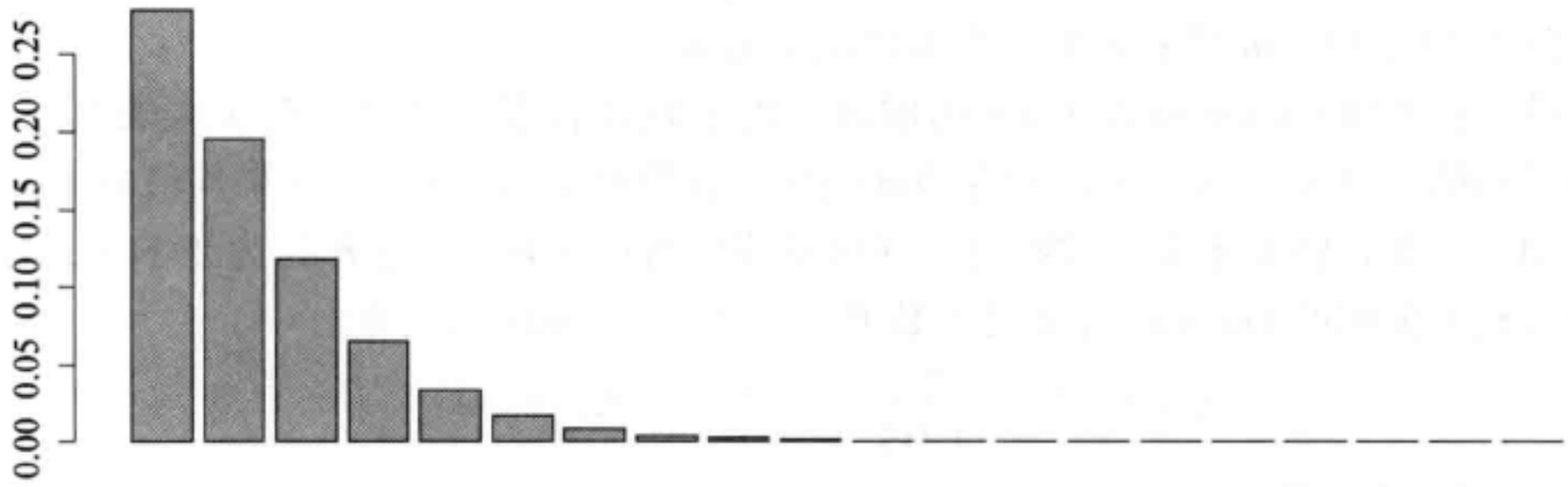


图 12-3 购买频率直方图

3. 平均单次购买金额的概率密度

根据假设五和假设六可以推导出顾客各期平均单次购买金额的概率密度函数 Gamma-Gamma 混合型函数（Colombo & Jiang 于 1999 年提出）：

$$g_{G\text{-}G}(m|u,v,\phi) = \frac{\Gamma(u+v)}{\Gamma(u)\Gamma(v)}\left(\frac{m}{\phi+m}\right)^{u}\left(\frac{\phi}{\phi+m}\right)^{v}\frac{1}{m} \tag{12-6}$$

由于购买金额 m 是一个连续型变量，所以公式 12-6 是平均单次购买金额的概率密度函数。R 语言代码如下：

```
prob.m<-function(m,u,v,o){
    result<-gamma(u+v)/(gamma(u)*gamma(v))
    result<-result*(m/(m+o))^u
    result<-result*(o/(m+o))^v
    result<-result/m
    result
}
```

假设 u=8.57,v=1.53,o=3224.15，则绘制 0 ～ 100000 的概率密度曲线的代码如下：

```
u<-8.57;v<-1.53; o<-3224.15
m<-c()
for(i in 1:100000){
    m[i]<-prob.m(i,u,v,o)
}
plot(m,type="l")
```

绘制结果如图 12-4 所示。

利用 bayes 理论可以预测用户 i 在下一周期的平均购买金额 m_i 的期望值 $E(m_i)$，公式如下：

$$E(m_i) = \frac{u(x_i\, m_i+\phi)}{(ux_i + v - 1)}$$

其中参数 u、v 和 ϕ 是公式 12-6 中的参数，x_i 是指顾客在上一个周期内的购买次数，$\overline{m}_i$

是顾客在上一周期的 *xi* 次交易中的平均购买金额。

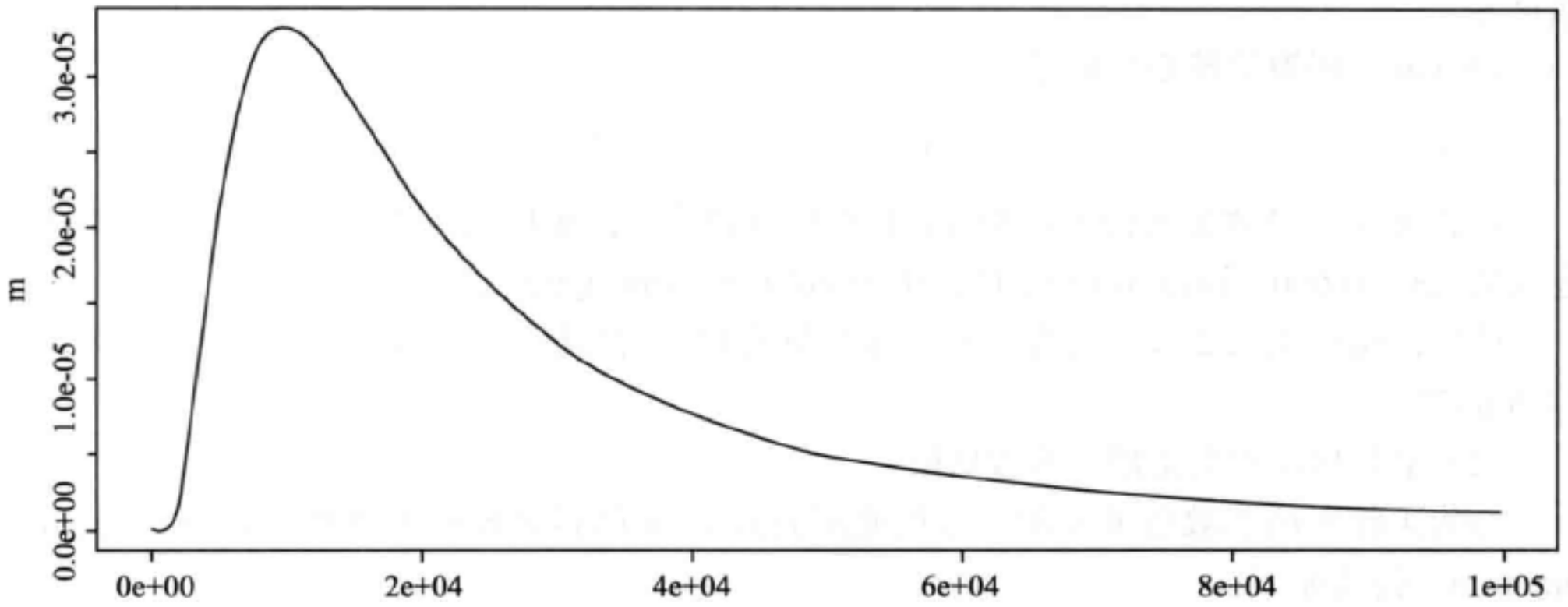


图 12-4 平均购买金额概率密度曲线

4. 使用极大似然法进行参数估值

（1）购买频率的参数估值

使用公式 12-5 得到对数极大似然函数（log-likelihood function），其公式如下：

$$LL(\alpha,w|f_1,f_2,\cdots,f_n)=\sum_{i=1}^{n}1n[\frac{\Gamma(w+f)}{\Gamma(w)f!}\left(\frac{\alpha}{\alpha+1}\right)^{w}\left(\frac{1}{\alpha+1}\right)^{f}]$$

其中，f_1，f_2,$\cdots f_n$ 是数据集中的购买频率 F 的样本。当对数极大似然函数取最大值时，对应的参数 α、w 是最优的参数估值。在 R 语言中，使用 nlminb 函数可以计算函数取最小值时参数的估值，最终实现代码如下：

```
a.w.ll<-function(params,data){
    f<-c()
    for(i in 1:length(data)){
        f[i]<-gamma(params[2]+data[i])*(params[1]/(params[1]+1))^params[2]
            *(1/(params[1]+1))^data[i]/(gamma(params[2])* factorial(data[i]))
    }
    ll<-sum(log(f))#log-likelihood 函数
    -ll
}
a.w.value<-function(F,params.upper=c(10,10)){
    geyser.res<-nlminb(c(0,0),a.w.ll,data=F,lower=c(0.00001,0.00001),upper=para
    ms.upper)
    params<-geyser.res$par
    a<-params[1];w<-params[2]
    c(a,w)
}
params.f<-a.w.value(data$F,c(20,20)); params.f
[1] 0.5453956    2.0906375
```

其中自定义的 a.w.ll 函数主要用于构建对数极大似然函数（Log-Likelihood Function）。

由于 nlminb() 函数用于最小化一个函数的值，所以输出前要在对数极大似然函数 ll 前添加负号。

a.w.value 函数的核心代码是：

```
nlminb(c(0,0),a.w.ll,data=F,lower=c(0.00001,0.00001),upper=params.upper)
```

其意义是：对数极大似然函数 a.w.ll 中样本值为 F，参数初始值为 0，两个参数估值的范围依次是 [0.0001, params.upper[1]] 和 [0.0001, params.upper[2]]。

最后，通过 12.2.2 节中的数据集 data 计算参数 a 的估值，该值为 0.5453956，w 估值为 2.0906375。

（2）平均单次购买金额的参数估值

与购买频率的参数估值类似，这里使用公式 12-6 得到对数极大似然函数 log-likelihood function，公式如下：

$$LL(u,v\phi \mid m_1, m_2, \cdots, m_n)=\sum_{i=1}^{n}\ln[\frac{\Gamma(u+v)}{\Gamma(u)\Gamma(v)}\left(\frac{m}{\phi+m}\right)^{u}\left(\frac{\phi}{\phi+m}\right)^{v}\frac{1}{m}]$$

其中，$m_1,m_2,\cdots m_n$ 是数据集中的购买频率 F 的 n 个样本。实现代码如下：

```
u.v.o.ll<-function(params,data){
    f<-gamma(params[1]+params[2])*(data/(data+params[3]))^params[1]
      *(params[3]/(data+params[3]))^params[2]/(gamma(params[1])*gamma(params[2]
      )*data)
    -sum(log(f))
}
u.v.o.value<-function(M,params.upper=c(20,100,20000)){
    geyser.res<-nlminb(c(0,0,0),u.v.o.ll,data=M,
               lower=c(0.00001,0.00001,0.00001),upper=params.upper)
    params<-geyser.res$par
    u<-params[1];v<-params[2];o<-params[3]
    c(u,v,o)
}
params.m<-u.v.o.value(data$M,c(20,100,20000)); params.m
[1]   3.937972    3.441782    447.964773
```

最后，通过 12.2.2 节中的数据集 data 计算参数 u 估值，该值为 3.937972，v 估值为 3.441782，o 估值为 447.964773。

12.3 结合马尔科夫链的 RFM 模型

在 12.2 节中介绍了用户购买频率和平均购买金额的模型，并介绍了预测下一周期的方法。但是该模型的缺点是只能通过孤立地使用购买频率或平均购买金额的历史数据来预测对应的指标。结合马尔科夫链的 RFM 模型则可以综合考量 3 个指标，并给出最佳预测值，其模型也相对健全。

12.3.1 购买状态转移矩阵

1. 一般结构

依据 12.2.1 节中介绍的 RFM 概念，可以总结出购买状态转移矩阵的一般结构，如图 12-5 所示。

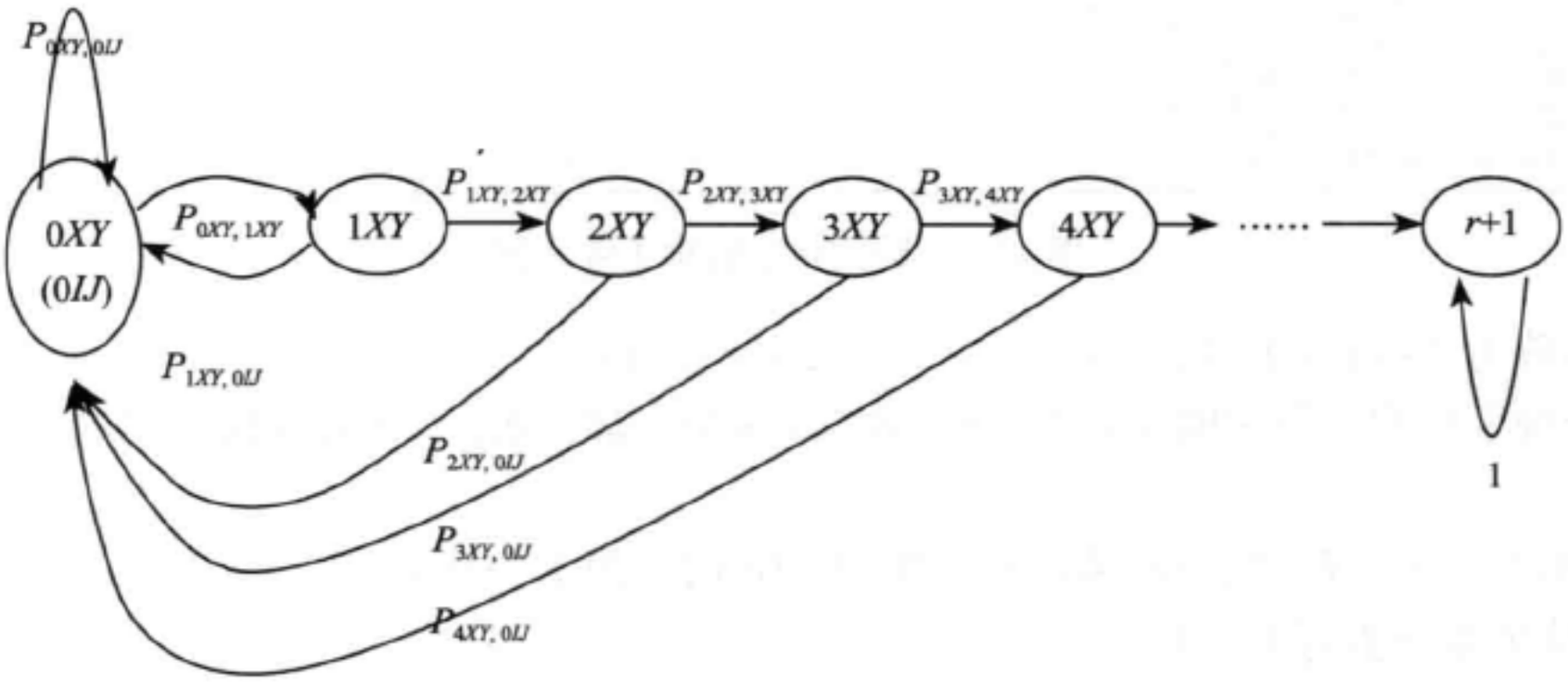


图 12-5 购买状态转移矩阵的一般结构

在图 12-5 中，$P(ijk, lmn)$ 代表当顾客这期的购买状态 (R, F, M) 为 (i, j, k) 时，下一期会转移到状态为 (l, m, n) 的概率。例如，假设本期顾客曾购买商品，则其在本期的状态为 $(R, F, M) = (0, X, Y)$，如果顾客在下一期购买了产品，而且其购买频率为 I，平均单次购买金额状态为 J，则其在下一期的状态由 $(0, X, Y)$ 转移到 $(0, I, J)$。如果顾客在下一期没有购买，则其状态由 $(0, X, Y)$ 转移到 $(1, X, Y)$。如果最近购买日 (R) 的取值为 $0 \sim r$（共 r+1 个），购买频率 (F) 的取值为 $1 \sim f$（共 f 个），则购买金额 (M) 经过离散化处理后可以取 $1 \sim m$ 个不同的购买状态（共 m 个）。注意，购买状态 (i, X, Y) 可以转移到任意 F、M 的组合（i, I, J）状态（共 $f \times m$ 种状态），并且 (i, X, Y) 也可以转移到 $(i$+1, $X, Y)$，即每个购买状态可以转移到 $f \times m$+1 种状态。一个购买状态转移矩阵为 $(r$+1) $\times f \times m$+1 行乘以 $(r$+1) $\times f \times m$+1 列。

假设 r=2，f=2，m=2，则图 12-5 可以简化为图 12-6。

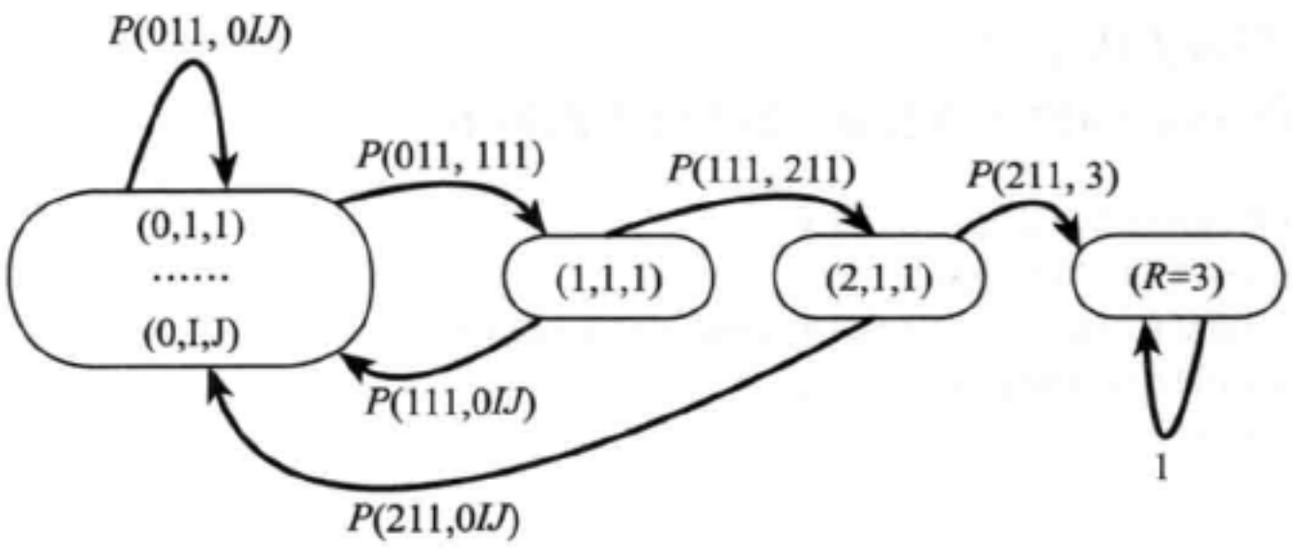


图 12-6 r=2，f=2，m=2 的马尔科夫链示例

在图 12-6 中，R=3 代表当 R=r+1=2+1=3 时的购买状态，它实际代表了商户处于流失状态不再购买的含义。对应的购买状态转移矩阵如图 12-7 所示。

	R=0, F=1, M=1	R=0, F=1, M=2	R=0, F=2, M=1	R=0, F=2, M=2	R=1, F=1, M=1	R=1, F=1, M=2	R=1, F=2, M=1	R=1, F=2, M=2	R=2, F=1, M=1	R=2, F=1, M=2	R=2, F=2, M=1	R=2, F=2, M=2	R=3
R=0, F=1, M=1	P(011, 011)	P(011, 012)	P(011, 021)	P(011, 022)	P(011, 111)	0	0	0	0	0	0	0	0
R=0, F=1, M=2	P(012, 011)	P(012, 012)	P(012, 021)	P(012, 022)	0	P(012, 112)	0	0	0	0	0	0	0
R=0, F=2, M=1	P(021, 011)	P(021, 012)	P(021, 021)	P(021, 022)	0	0	P(021, 121)	0	0	0	0	0	0
R=0, F=2, M=2	P(022, 011)	P(022, 012)	P(022, 021)	P(022, 022)	0	0	0	P(022, 122)	0	0	0	0	0
R=1, F=1, M=1	P(111, 011)	P(111, 012)	P(111, 021)	P(111, 022)	0	0	0	0	P(111, 211)	0	0	0	0
R=1, F=1, M=2	P(112, 011)	P(112, 012)	P(112, 021)	P(112, 022)	0	0	0	0	0	P(112, 212)	0	0	0
R=1, F=2, M=1	P(121, 011)	P(121, 012)	P(121, 021)	P(121, 022)	0	0	0	0	0	0	P(121, 221)	0	0
R=1, F=2, M=2	P(122, 011)	P(122, 012)	P(122, 021)	P(122, 022)	0	0	0	0	0	0	0	P(122, 222)	0
R=2, F=1, M=1	P(211, 011)	P(211, 012)	P(211, 021)	P(211, 022)	0	0	0	0	0	0	0	0	P(211, 3)
R=2, F=1, M=2	P(212, 011)	P(212, 012)	P(212, 021)	P(212, 022)	0	0	0	0	0	0	0	0	P(212, 3)
R=2, F=2, M=1	P(221, 011)	P(221, 012)	P(221, 021)	P(221, 022)	0	0	0	0	0	0	0	0	P(221, 3)
R=2, F=2, M=2	P(222, 011)	P(222, 012)	P(222, 021)	P(222, 022)	0	0	0	0	0	0	0	0	P(222, 3)
R=3	0	0	0	0	0	0	0	0	0	0	0	0	1

图 12-7 购买状态转移矩阵示例

该矩阵为 13 行 ×13 列，其中（2+1）×2×2+1=13。

由于前面已假设顾客购买频率（F）和购买金额（M）独立，因此在接下来的章节中将分开讨论：

1）购买频率（F）状态转移频率：在 M 固定时，由上一期 R=r1，F=f1 的购买状态计算本期 F=f2 的概率 $f_f(f_2|f_1, r_1)$。

2）购买金额（M）状态转移频率：在 F 和 R 固定时，由上一期 M=m1(a1<m1<b1) 的购买状态计算本期 M=m2 的概率 $f_m(m_2|a_1<m_1<b_1)$。

然后结合两者完整地计算顾客 RFM 购买状态转移机率，最后讨论特殊情况下的状态转移概率。

2. 购买频率状态转移概率

在 M 固定时，由上一期 R=r1，F=f1 的购买状态计算本期 F=f2 的概率 $f_f(f_2|f_1,r_1)$，公式如下：

$$f_f(f_2|f_1,r_1) = \frac{(1+r_1+\alpha)^{f_1+w}}{\Gamma(f_1+w)f_2!}\frac{\Gamma(f_2+f_1+w)}{(2+r_1+\alpha)^{f2+f1+w}} \tag{12-7}$$

假设起始状态为 R=r1 和 F=f1 时，下一期会转移到 (R, F)=(0, f2) 的概率为 $f_f(f_2|f_1, r_1)$，此时表示下一期顾客发生购买行为；而当下一期顾客没有发生购买行为时，顾客转移到 (R, F)=(1+r1, f1) 的概率为 $f_f(0|f_1, r_1)$。

使用 gamma 和 prod 函数实现公式 12-7 的代码如下：

```
prob.ff<-function(f1,r1,f2,a,w){
    result<-(1+r1+a)^(f1+w)
    result<-result/(gamma(w+f1)*prod(c(1:f2)))
    result<-result*gamma(w+f1+f2)
    result<-result/((2+r1+a)^(f1+f2+w))
    result
}
```

这里自定义了一个 prob.ff 函数用于实现公式 12-7。

3. 购买金额状态转移概率

F 和 R 固定时，如果观察到顾客前期的购买金额 $m1$ 属于某一购买金额状态（假设此购买金额状态范围为 $a1<m1<b1$），那么下一期购买金额的事后概率函数如下：

$$f_m(m|a_1<m_1<b_1)=\frac{\int_{a_1}^{b_1}\frac{m^{u-1}(\phi+m_1)^{u+v}\Gamma(2u+v)}{\Gamma(u)\Gamma(u+v)(m_1+m+\phi)^{2u+v}}Wdm_1}{\int_{a_1}^{b_1}\frac{\Gamma(u+v)}{\Gamma(u)\Gamma(v)}\left(\frac{m}{(\phi+m)}\right)^u\left(\frac{\phi}{(\phi+m)}\right)^v\frac{1}{m}dm}$$

$$\text{其中 } W=\left[\frac{\Gamma(u+v)}{\Gamma(u)\Gamma(v)}\left(\frac{m_1}{(\phi+m_1)}\right)^u\left(\frac{\phi}{(\phi+m_1)}\right)^v\frac{1}{m_1}\right] \tag{12-8}$$

由公式 12-8 可以推导出：如果顾客前期购买金额状态为 $a1<m1<b1$，那么此顾客下一期购买金额 $m2$ 转移至某一购买金额状态（假设此购买金额状态范围为 $a2<m2<b2$）的事后概率函数如下：

$$\begin{aligned}&f_m(a_2<m_2<b_2|a_1<m_1<b_1)\\&=\int_{a_2}^{b_2}f_m(m_2|a_1<m_1<b_1)dm_2\end{aligned} \tag{12-9}$$

把公式 12-8 代入公式 12-9 中，得到的完整公式为：

$$f_m(a_2<m_2<b_2|a_1<m_1<b_1)=\int_{a_2}^{b_2}f_m(m_2|a_1<m_1<b_1)dm_2$$

$$=\int_{a_2}^{b_2}\frac{\int_{a_1}^{b_1}\frac{m^{u-1}(\phi+m_1)^{u+v}\Gamma(2u+v)}{\Gamma(u)\Gamma(u+v)(m_1+m+\phi)^{2u+v}}Wdm_1}{\int_{a_1}^{b_1}\frac{\Gamma(u+v)}{\Gamma(u)\Gamma(v)}\left(\frac{m}{(\phi+m)}\right)^u\left(\frac{\phi}{(\phi+m)}\right)^v\frac{1}{m}dm}dm_2$$

$$=\int_{a_2}^{b_2}\frac{\int_{a_1}^{b_1}\frac{m^{u-1}(\phi+m_1)^{u+v}\Gamma(2u+v)}{\Gamma(u)\Gamma(u+v)(m_1+m+\phi)^{2u+v}}\frac{\Gamma(u+v)\left(\frac{m_1}{(\phi+m_1)}\right)^u\left(\frac{\phi}{(\phi+m_1)}\right)^v}{\Gamma(u)\Gamma(v)\,m_1}dm_1}{\int_{a_1}^{b_1}\frac{\Gamma(u+v)}{\Gamma(u)\Gamma(v)}\left(\frac{m}{(\phi+m)}\right)^u\left(\frac{\phi}{(\phi+m)}\right)^v\frac{1}{m}dm}dm_2$$

（12-10）

实现公式 12-10 的步骤如下：

1）计算：

$$f.\text{deno}(a1,b1,u,v,o)=\int_{a_1}^{b_1}\frac{\Gamma(u+v)}{\Gamma(u)\Gamma(v)}\left(\frac{m}{(\phi+m)}\right)^u\left(\frac{\phi}{(\phi+m)}\right)^v\frac{1}{m}dm$$

已知 $a1$、$b1$、u、v、o，使用积分函数 integrate 实现的代码如下：

```
denominator<-integrate(
                function(x)  (gamma(u+v)*(x/(o+x))^u*(o/(o+x))^v)/
                (gamma(u)*gamma(v)*x),
                lower=a1,upper=b1) #x 为上面公式中的 m
denominator<-denominator$value
```

其中 integrate 函数的第一个参数为欲积分的一元函数（这里的积分变量为 x），lower 和 upper 是积分的上限和下限。

2）把计算好的数值代入公式 12-10 中，简化为：

$$f_m(a_2<m_2<b_2|a_1<m_1<b_1)=\int_{a_2}^{b_2}\left(\int_{a_1}^{b_1}\frac{m^{u-1}(\phi+m_1)^{u+v}\Gamma(2u+v)}{\Gamma(u)\Gamma(u+v)(m_1+m+\phi)^{2u+v}}\;\frac{\Gamma(u+v)\left(\frac{m_1}{(\phi+m_1)}\right)^u\left(\frac{\phi}{(\phi+m_1)}\right)^v}{\Gamma(u)\Gamma(v)\,m_1\;f.\mathrm{deno}(a1,b1,u,v,o)}\,dm_1\right)dm_2$$

它实际是对函数 $\frac{m^{u-1}(\phi+m_1)^{u+v}\Gamma(2u+v)}{\Gamma(u)\Gamma(u+v)(m_1+m+\phi)^{2u+v}}\;\frac{\Gamma(u+v)\left(\frac{m_1}{(\phi+m_1)}\right)^u\left(\frac{\phi}{(\phi+m_1)}\right)^v}{\Gamma(u)\Gamma(v)\,m_1\;f.\mathrm{deno}(a1,b1,u,v,o)}$ 先对变量 m_1 后对变量 m_2 的二重积分，积分上下限分别为 $a1,b1$ 和 $a2,b2$。使用 pracma 扩展包中的二重积分函数 integral2 实现的代码如下：

```
result<-integral2(
            function(x,y)
            (y^(u-1)*(o+x)^(u+v)*gamma(2*u+v)*gamma(u+v)*(x/(o+x))^u*(o/
            (o+x))^v)
            /(denominator*gamma(u)*gamma(u+v)*(x+y+o)^(2*u+v)*gamma(u)*gamma(v)
            *x)
            ,xmin=a1,xmax=b1,ymin=a2,ymax=b2)#x 为上面公式中的 m1，y 为 m2
result$Q
```

result$Q 为计算公式 12-9 的最终结果，把上述代码封装成自定义函数 prob.fm 的代码如下：

```
library("pracma")
prob.fm<-function(a1,b1,a2,b2,u,v,o){
    denominator<-integrate(
                function(x)  (gamma(u+v)*(x/(o+x))^u*(o/(o+x))^v)/
                (gamma(u)*gamma(v)*x)
                ,lower=a1,upper=b1)
    denominator<-denominator$value
    result<-integral2(
                function(x,y)
                (y^(u-1)*(o+x)^(u+v)*gamma(2*u+v)*gamma(u+v)*(x/(o+x))^u*(o/
                (o+x))^v)
                /(denominator*gamma(u)*gamma(u+v)*(x+y+o)^(2*u+v)*gamma(u)*gamm
                a(v)*x)
                ,xmin=a1,xmax=b1,ymin=a2,ymax=b2)
    result$Q
}
```

4. RFM 整体状态转移概率

根据购买频率和金额的状态转移函数，以及顾客的购买频率和金额独立的假设，可以推导出顾客 RFM 状态的转移概率函数相关公式。

（1）顾客下一期没有发生购买行为时

当顾客下一期没有发生购买行为时，顾客购买状态从 (R,F,M)=(r_1,f_1,a_1<m_1<b_1) 转移至 (r_1+1,f_1,a_1<m_1<b_1)，公式如下：

$$\begin{aligned} & f_{r,f,m}(r_1+1, f_1, a_1<m_1<b_1 | r_1, f_1, a_1<m_1<b_1) \\ & =f_f(0 \mid f_1, r_1) \end{aligned} \tag{12-11}$$

（2）顾客下一期发生购买行为时

当顾客下一期发生购买行为时，顾客购买状态从 (R,F,M)=(r_1,f_1,a_1<m_1<b_1) 转移至 (0,f_2,a_2<m_2<b_2)，公式如下：

$$\begin{aligned} & f_{r,f,m}(0, f_2, a_2<m_2<b_2 | r_1, f_1, a_1<m_1<b_1) \\ & =f_f(f_2 \mid f_1, r_1) f_m(a_2<m_2<b_2 \mid a_1<m_1<b_1) \end{aligned} \tag{12-12}$$

5. RFM 特殊状态转移概率

在定义购买频率状态时，必须考虑如何定义最后一个购买频率的状态。如果将最后一个状态定义为某一购买频率 f，即假设顾客购买频率的最大值为 f，则此为不合理的假设，可以将购买频率最后一个状态 f 定义为：购买频率 >f。由最后一个状态到其他状态的购买频率 f_f 的公式如下：

$$\begin{aligned} f_f(f_2 \mid r_1, f_1 \geqslant f) &= \frac{P(f_2, r_1, f_1 \geqslant f)}{P(r_1, f_1 \geqslant f)} = \frac{\sum_{f_1=f}^{\infty} f_f(f_2 \mid r_1, f_1) f_{r,f}(r_1, f_1)}{\sum_{f_1=f}^{\infty} f_{r,f}(r_1, f_1)} \\ &= \frac{\sum_{f_1=f}^{\infty} \left[\frac{(1+r_1+\alpha)^{f_1+w} \Gamma(f_2+f_1+w)}{\Gamma(f_1+w) f_2!(2+r_1+\alpha)^{f_2+f_1+w}} \frac{\alpha^w \Gamma(f_1+w)}{\Gamma(w) f_1!(1+r_1+\alpha)^{f_1+w}} \right]}{\sum_{f_1=f}^{\infty} \left[\frac{\alpha^w \Gamma(f_1+w)}{\Gamma(w) f_1!(1+r_1+\alpha)^{f_1+w}} \right]} \end{aligned} \tag{12-13}$$

其实现代码如下：

```
prob.ff2.molecule<-function(f1,r1,f2,a,w){
    (1+r1+a)^(f1+w)*gamma(f1+f2+w)*a^w*gamma(f1+w)
/(gamma(f1+w)*factorial(f2)*(2+r1+a)^(f2+f1+w)*gamma(w)*factorial(f1)*(1+r1+a)^
(f1+w))
}
prob.ff2.denominator<-function(f1,r1,f2,a,w){
    a^w*gamma(f1+w)/(gamma(w)*factorial(f1)*(1+r1+a)^(f1+w))
}
#f.max 是公式 12-13 中的 f，f.inf 是虚拟的 inf 无穷大
```

```
prob.ff2<-function(r1,f.max,f.inf=50,f2,a,w){
    f1<-c(f.max:f.inf);molecule<-0;denominator<-0;
    for(i.f1 in f1){
        molecule<-molecule+prob.ff2.molecule(i.f1,r1,f2,a,w)
        denominator<-denominator+prob.ff2.denominator(i.f1,r1,f2,a,w)
    }
    molecule/denominator
}
```

prob.ff2.molecule 函数用于计算公式 12-13 中的分子：

$$\frac{(1+r_1+\alpha)^{f_1+w}\Gamma(f_2+f_1+w)}{\Gamma(f_1+w)f_2!(2+r_1+\alpha)^{f_2+f_1+w}}\quad\frac{\alpha^w\Gamma(f_1+w)}{\Gamma(w)f_1!(1+r_1+\alpha)^{f_1+w}}$$

prob.ff2.denominator 函数用于计算公式 12-13 中的分母：

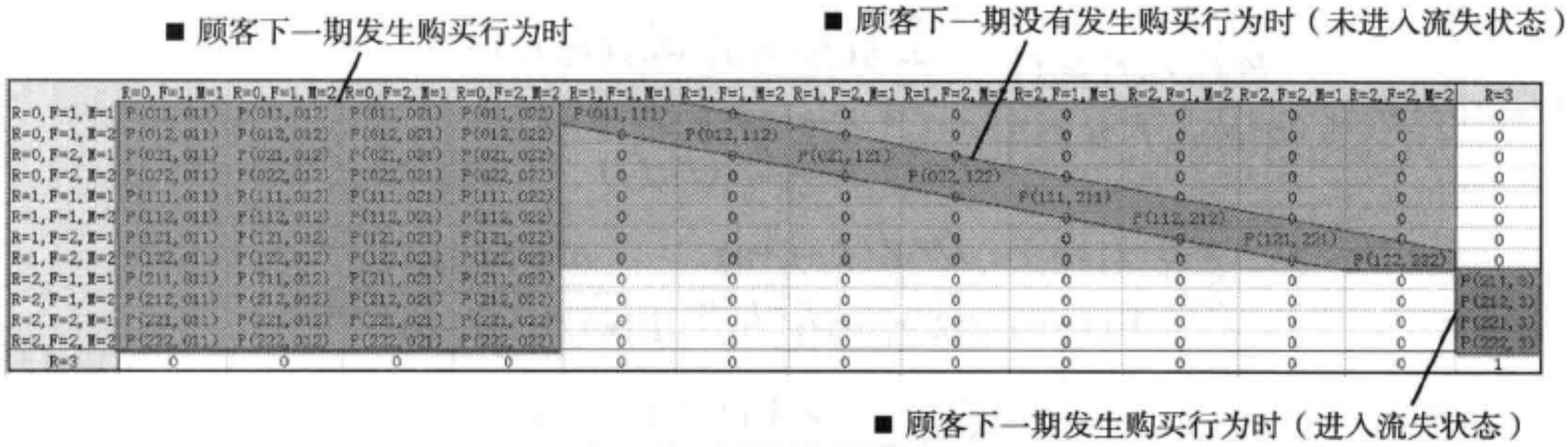
$$\frac{\alpha^w\Gamma(f_1+w)}{\Gamma(w)f_1!(1+r_1+\alpha)^{f_1+w}}$$

最后将上述函数统一封装进 prob.ff2 函数，用于无限累加。其中，参数 f.max 是公式 12-13 中的 f，参数 f.inf 是虚拟的 inf 无穷大。参数 f.inf 不要设置过高，可以设置为 f.max+20，设置过高会影响后续计算转移矩阵的时间。

6. 购买状态转移矩阵的 R 语言实现

（1）结构说明

为了简化问题，先讨论 r=2，f=2，m=2 时转移矩阵的情况，如图 12-8 所示。



	R=0,F=1,M=1	R=0,F=1,M=2	R=0,F=2,M=1	R=0,F=2,M=2	R=1,F=1,M=1	R=1,F=1,M=2	R=1,F=2,M=1	R=1,F=2,M=2	R=2,F=1,M=1	R=2,F=1,M=2	R=2,F=2,M=1	R=2,F=2,M=2	R=3
R=0,F=1,M=1	P(011,011)	P(011,012)	P(011,021)	P(011,022)	P(011,111)	0	0	0	0	0	0	0	0
R=0,F=1,M=2	P(012,011)	P(012,012)	P(012,021)	P(012,022)	0	P(012,112)	0	0	0	0	0	0	0
R=0,F=2,M=1	P(021,011)	P(021,012)	P(021,021)	P(021,022)	0	0	P(021,121)	0	0	0	0	0	0
R=0,F=2,M=2	P(022,011)	P(022,012)	P(022,021)	P(022,022)	0	0	0	P(022,122)	0	0	0	0	0
R=1,F=1,M=1	P(111,011)	P(111,012)	P(111,021)	P(111,022)	0	0	0	0	P(111,211)	0	0	0	0
R=1,F=1,M=2	P(112,011)	P(112,012)	P(112,021)	P(114,022)	0	0	0	0	0	P(112,212)	0	0	0
R=1,F=2,M=1	P(121,011)	P(121,012)	P(121,021)	P(121,022)	0	0	0	0	0	0	P(121,221)	0	0
R=1,F=2,M=2	P(122,011)	P(122,012)	P(122,021)	P(122,022)	0	0	0	0	0	0	0	P(122,222)	0
R=2,F=1,M=1	P(211,011)	P(211,012)	P(211,021)	P(211,022)	0	0	0	0	0	0	0	0	P(211,3)
R=2,F=1,M=2	P(212,011)	P(212,012)	P(212,021)	P(212,022)	0	0	0	0	0	0	0	0	P(212,3)
R=2,F=2,M=1	P(221,011)	P(221,012)	P(221,021)	P(221,022)	0	0	0	0	0	0	0	0	P(221,3)
R=2,F=2,M=2	P(222,011)	P(222,012)	P(222,021)	P(222,022)	0	0	0	0	0	0	0	0	P(222,3)
R=3	0	0	0	0	0	0	0	0	0	0	0	0	1

图 12-8 转移矩阵的结构说明

其中左侧的绿色区域表示顾客在下一期发生购买行为，可使用公式 12-11 计算。注意当上一期购买状态（见列坐标）的 F 值为最大值 2 时，计算绿色区域的该行元素 $P(ijk, lmn)$ 就需要先使用公式 12-13 代替公式 12-7，以便计算 f_f，然后结合公式 12-11 计算最后结果。

中间的红色区域和右下角的蓝色区域表示顾客下一期发生购买行为，可使用公式 12-12 计算。同样，当上一期购买状态（见列坐标）的 F 值为最大值 2 时，也要先使用公式 12-13 代替公式 12-7 后再结合公式 12-12 计算。

（2）划分购买状态

假设用户 10 个周期（相当于 20 个星期）没有购买商品，则表示不会再购买，即 R 的购买状态取值范围为 0 ～ 10。购买频率 F 的最大状态为数据集中 F 的最大值，即 F 的购买状态值为 1 ～ max(data$F)。实现代码如下：

```
num.f<-max(data$F)
num.r<-10
```

分析数据集中的 M 列数据，因其最大值为 max(data$M)=3005，所以可以把 M 分隔为 31 等份，每份的间隔为 100。即把连续变量的 M（其取值范围为 1 ～ 3005）分割为 (0,100], (100, 200],....,(3000,3100] 这 31 个等级（离散化），每个等级对应的范围下限 $m.a$ 和上限 $m.b$ 依次为：

```
num.m<-31
m.a<-c(0:30)*100#m 的下限 (0,100],(100,200],....(3000,3100] 分别为 1---31
m.b<-c(1:31)*100#m 的上限 (0,100],(100,200],....(3000,3100] 分别为 1---31
```

（3）计算状态转移矩阵

在计算状态转移矩阵前，要先设置购买频率的概率参数 a 和 w，该参数的估值已经在 12.2.3 节中使用极大似然法计算完成，并存储于 params.f 中。设置平均单次购买金额的概率参数 u、v 和 o，同样，在 12.2.3 节中也给出了计算结果 params.m。最后将用于计算特殊状态转移概率模拟无穷大时的参数 f.inf（用于实现公式 12-13 的 prob.ff2 函数）设置为 50。实现代码如下：

```
f.inf<-50
a<-params.f[1];w<-params.f[2]
u<-params.m[1];v<-params.m[2];o<-params.m[3]
```

建立一个 (r.num+1) × f.num × m.num +1 行乘以 (r.num +1) × f.num × m.num +1 列的矩阵，并以此计算不为 0 的 $P(ijk, lmn)$ 元素。其中 $P(ijk, lmn)$ 代表当顾客这期的购买状态 (R, F, M) 为 (i, j, k) 时，下一期转移到状态为 (l,m,n) 的概率。此时元素行坐标的计算公式为 i*(num.f*num.m)+(j−1)*num.m+k，元素列坐标的计算公式为 l*(num.f*num.m)+(m−1)*num.m+n。代码如下：

```
p<-matrix(0,ncol=(num.r+1)*num.f*num.m+1,nrow=(num.r+1)*num.f*num.m+1)
p[ncol(p),nrow(p)]<-1
for(i.m in 1:num.m){
  for(i.f in 1:num.f){
    if(i.f!=num.f){
        #使用公式 12-11 计算
        p[num.r*(num.f*num.m)+(i.f-1)*num.m+i.m,nrow(p)]<-
                    prob.ff(f1=i.f,r1=num.r,f2=0,a=a,w=w)
    }else{
        #使用公式 12-13 计算
        p[num.r*(num.f*num.m)+(i.f-1)*num.m+i.m,nrow(p)]<-
                    prob.ff2(r1=num.r,f.max=num.f,f.inf=f.inf,f2=0,a=a,w=w)
}}}
```

```
for(i.r in 0:num.r){
  for(i.m in 1:num.m){
    for(i.f in 1:num.f){
      for(i.m.now in 1:num.m){
        for(i.f.now in 1:num.f){
          if(i.f!=num.f){
              #使用公式 12-12 计算
              p[i.r*(num.f*num.m)+(i.f-1)*num.m+i.m,(i.f.now-1)*num.m+i.
              m.now]<-
                    prob.ff(f1=i.f,r1=i.r,f2=i.f.now,a=a,w=w)*
                    prob.fm(a1=m.a[i.m],b1=m.b[i.m],a2=m.a[i.m.now],b2=m.b[i.
                    m.now],u=u,v=v,o=o)
          }else{
              #使用公式 12-13 代替公式 12-7 后，结合公式 12-13 计算
              p[i.r*(num.f*num.m)+(i.f-1)*num.m+i.m,(i.f.now-1)*num.m+i.
              m.now]<-
                    prob.ff2(r1=i.r,f.max=num.f,f.inf=f.inf,f2=i.
                    f.now,a=a,w=w)*
                    prob.fm(a1=m.a[i.m],b1=m.b[i.m],a2=m.a[i.m.now],b2=m.b[i.
                    m.now],u=u,v=v,o=o)
          }}}
    if(i.r!=num.r){
      if(i.f!=num.f){
        #使用公式 12-11 计算
        p[i.r*(num.f*num.m)+(i.f-1)*num.m+i.m,(i.r+1)*(num.f*num.m)+(i.
        f-1)*num.m+i.m]<-
              prob.ff(f1=i.f,r1=i.r,f2=0,a=a,w=w)
      }else{
        #使用公式 12-13 计算
        p[i.r*(num.f*num.m)+(i.f-1)*num.m+i.m,(i.r+1)*(num.f*num.m)+(i.
        f-1)*num.m+i.m]<-
              prob.ff2(r1=i.r,f.max=num.f,f.inf=f.inf,f2=0,a=a,w=w)
}}}}}
```

最后得到的矩阵 p 就是购买状态转移矩阵。下面把矩阵 p 每行累加后观察其数值分布：

```
hist(apply(p,1,sum))
```

结果如图 12-9 所示。

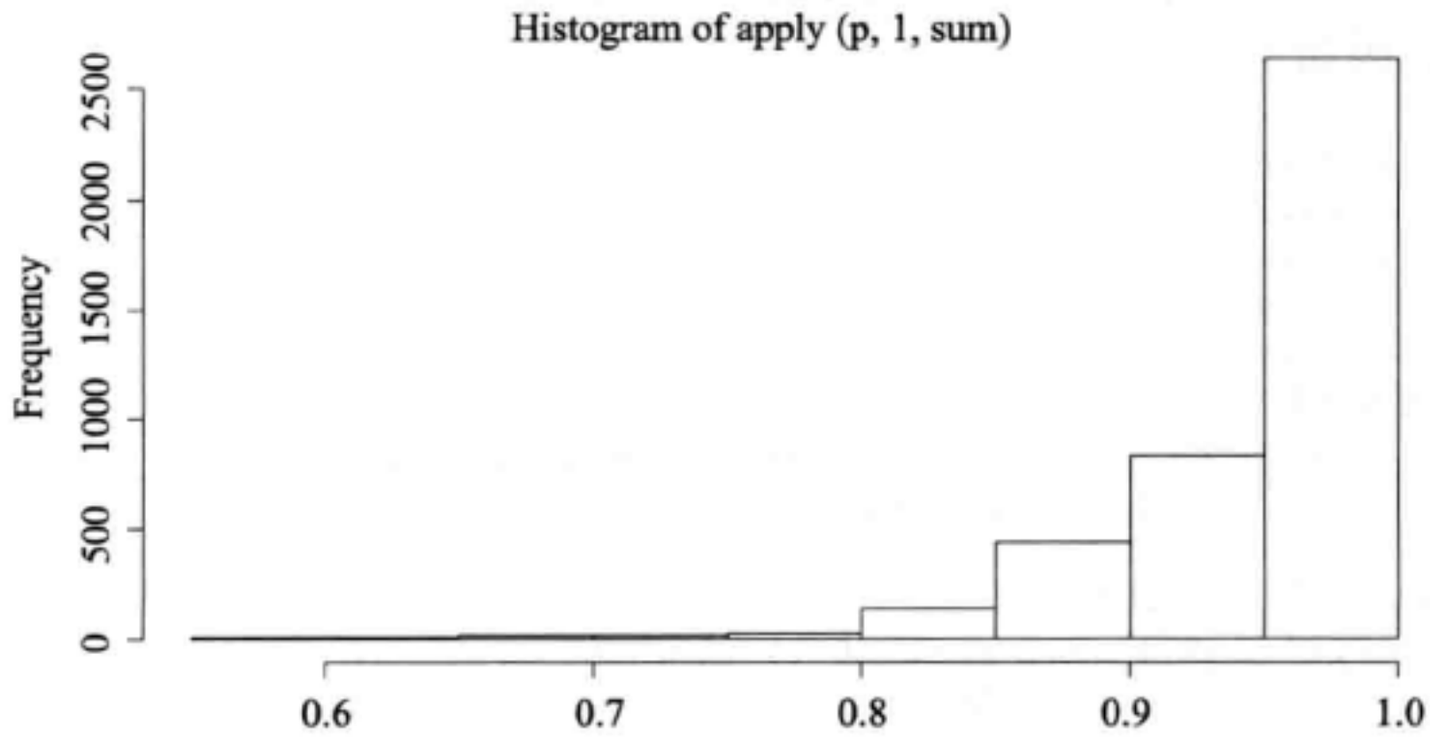


图 12-9 结果展示

理论上转移矩阵每行的累加和应该为 1，而计算的转移矩阵 p 由于有些数值计算误差（模拟无穷大求和以及模拟积分运算等都会出现误差），导致其行累加并未均为 1。

12.3.2 预测用户购买状态

得到转移矩阵后，可以采用类似 12.1.3 节介绍的方法来预测某用户后续的购买状态。例如，在数据集中，某用户在 2012.2.6—2012.2.19 内的 RFM 数据 (*R*,*F*,*M*) 为 (8, 4, 569.24)，根据 *M* 的分割规则，其离散化后的状态为 (8,4,6)。其购买状态可以使用只有一行的矩阵 1.0 表示，其中第 8*(num.*f*num.*m*)+(4-1)*num.*m*+6 个元素为 1，其他均为 0。代码如下：

```
num<-(num.r+1)*num.f*num.m+1
l.0<-matrix(rep(0,num),1)
l.0[1, 8*(num.f*num.m)+(4-1)*num.m+6]<-1
```

可预测在接下来的两个星期内（1 个 RFM 统计周期）的购买状态为：

```
l.1<- l.0%*%p
```

最后预测下个周期的平均购买金额，首先建立一个系数矩阵，当 *R* 取值不等于 0 时，表示预测下一周期并未发生购买行为，系数矩阵的对应元素值为 0。否则当 *R* 取值等于 0，且 *F* 取值为 *i.f* 状态时，系数矩阵对应的元素值为 (*i.f*-1) × 100+50.5。最后与 l.1 矩阵相乘，得到预测的平均购买金额。代码如下：

```
l<-matrix(rep(0,num),num)
for(i.r in 0:num.r){
  for(i.m in 1:num.m){
    for(i.f in 1:num.f){
       i<- i.r*(num.f*num.m)+(i.f-1)*num.m+i.m
       if(i.r!=0){
          l[i,1]<-0
       }else{
          l[i,1]<-(i.f-1)*100+50.5
}}}}
```

得到系数矩阵 l 后，可以预测未来 40 个周期（80 个星期）后的用户平均购买金额。通过统计 *R* 等于 0 的概率来预测用户下一期发生购买行为的概率。代码如下：

```
m<-c()
for(i in 1:40){
    if(i==1){
        l.tmp<-l.0%*%p
}else{
        l.tmp<-l.tmp%*%p
    }
    m[i]<-as.numeric(l.tmp%*%l)
}
```

然后绘制曲线图，代码如下：

```
plot(m,type="b")
```

绘制结果如图 12-10 所示。

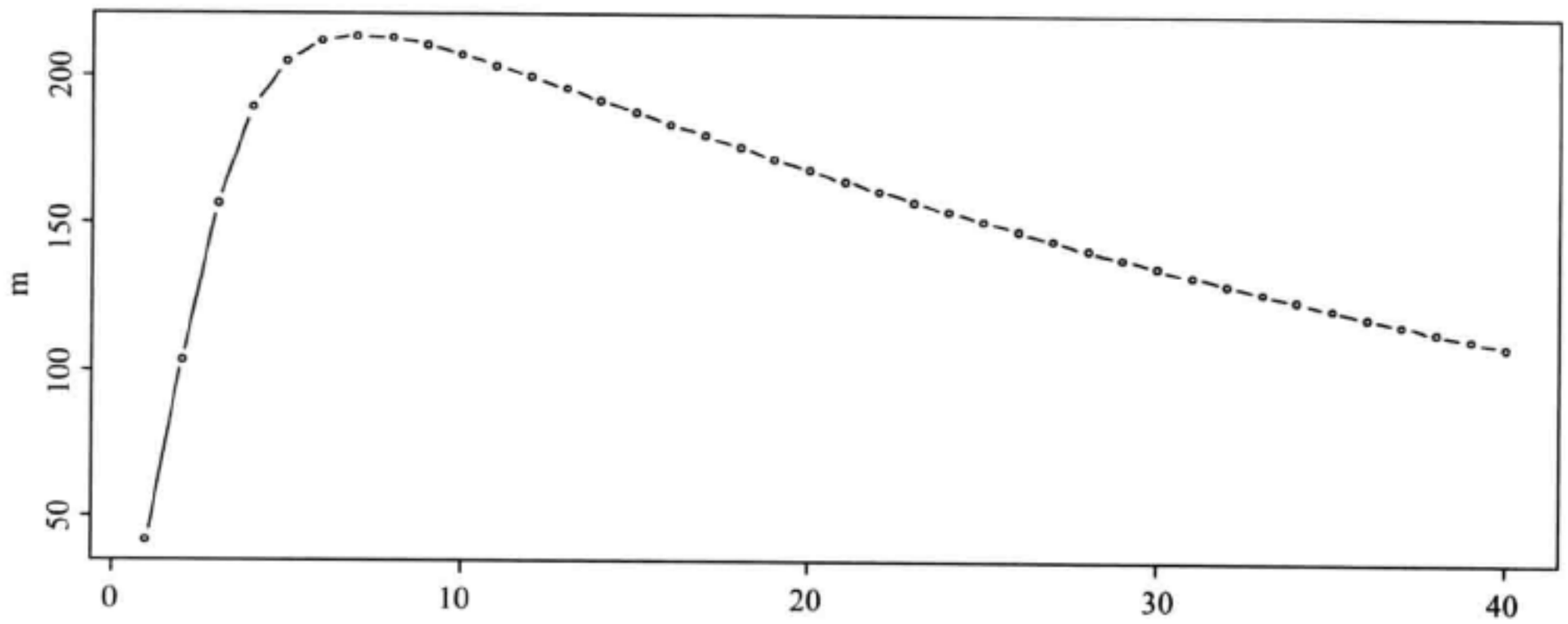


图 12-10　未来 40 个周期的平均购买金额曲线图

注意图 12-10 并非表示该用户会在接下来的 40 个周期内均会购买相应消费金额的商品。它只表示在接下来的 40 个周期内发生购买行为时，其对应的平均消费金额。例如 m[1] 等于 41.21216 表示如果用户在下一个周期购买了商品，则其平均消费金额最大可能性为 41.21216。另外，可以看到，随着时间的变化，预测的平均消费金额逐步上升。用户如果在接下来的第 7 个周期购买了商品，则其平均消费金额会最高，约为 213.95599。接下来预测的平均消费金额会逐步下降。